

SemEval 2021

**The 15th International Workshop  
on Semantic Evaluation (SemEval-2021)**

**Proceedings of the Workshop**

August 5 - 6, 2021  
Bangkok, Thailand (online)

©2021 The Association for Computational Linguistics  
and The Asian Federation of Natural Language Processing

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)  
209 N. Eighth Street  
Stroudsburg, PA 18360  
USA  
Tel: +1-570-476-8006  
Fax: +1-570-476-0860  
[acl@aclweb.org](mailto:acl@aclweb.org)

ISBN 978-1-954085-70-1

# Introduction

Welcome to SemEval-2021!

The Semantic Evaluation (SemEval) series of workshops focuses on the evaluation and comparison of systems that can analyze diverse semantic phenomena in text, with the aims of extending the current state of the art in semantic analysis and creating high quality annotated datasets in a range of increasingly challenging problems in natural language semantics. SemEval provides an exciting forum for researchers to propose challenging research problems in semantics and to build systems/techniques to address such research problems.

SemEval-2021 is the fifteenth workshop in the series of International Workshops on Semantic Evaluation. The first three workshops, SensEval-1 (1998), SensEval-2 (2001), and SensEval-3 (2004), focused on word sense disambiguation, each time expanding in the number of languages offered, the number of tasks, and also the number of teams participating. In 2007, the workshop was renamed to SemEval, and the subsequent SemEval workshops evolved to include semantic analysis tasks beyond word sense disambiguation. In 2012, SemEval became a yearly event. It currently takes place every year, on a two-year cycle. The tasks for SemEval-2021 were proposed in 2020, and next year's tasks have already been selected and are underway.

SemEval-2021 is co-located (virtually) with The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021) on August 5–6. This year's SemEval included the following 11 tasks:

- Lexical semantics
  - Task 1: Lexical Complexity Prediction
  - Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation
  - Task 4: Reading Comprehension of Abstract Meaning
- Social factors & opinion
  - Task 5: Toxic Spans Detection
  - Task 6: Detection of Persuasive Techniques in Texts and Images
  - Task 7: HaHackathon: Detecting and Rating Humor and Offense
- Information in scientific & clinical text
  - Task 8: MeasEval: Counts and Measurements
  - Task 9: Statement Verification and Evidence Finding with Tables
  - Task 10: Source-Free Domain Adaptation for Semantic Processing
  - Task 11: NLPContributionGraph
- Other phenomena
  - Task 12: Learning with Disagreements

This volume contains both task description papers that describe each of the above tasks and system description papers that present the systems that participated in the tasks. A total of 11 task description papers and 175 system description papers are included in this volume.

SemEval-2021 features two awards, one for organizers of a task and one for a team participating in a task. The Best Task award recognizes a task that stands out for making an important intellectual contribution to empirical computational semantics, as demonstrated by a creative, interesting, and scientifically rigorous dataset and evaluation design, and a well-written task overview paper. The Best Paper award recognizes a system description paper (written by a team participating in one of the tasks) that advances our understanding of a problem and available solutions with respect to a task. It need not be the highest-scoring system in the task, but it must have a strong analysis component in the evaluation, as well as a clear and reproducible description of the problem, algorithms, and methodology.

2021 has been another particularly challenging year across the globe. We are immensely grateful to the task organizers for their perseverance through many ups, downs, and uncertainties, as well as to the large number of participants whose enthusiastic participation has made SemEval once again a successful event! Thanks also to the task organizers who served as area chairs for their tasks, and to both task organizers and participants who reviewed paper submissions. These proceedings have greatly benefited from their detailed and thoughtful feedback. Thousands of thanks to our assistant organizers Julia R. Bonn and Abhidip Bhattacharyya for their extensive, detailed, and dedicated work on the production of these proceedings! We also thank the members of the program committee who reviewed the submitted task proposals and helped us to select this exciting set of tasks, and we thank the ACL 2021 conference organizers for their support. Finally, we most gratefully acknowledge the support of our sponsor: the ACL Special Interest Group on the Lexicon (SIGLEX).

The SemEval-2021 organizers: Guy Emerson, Aurelie Herbelot, Alexis Palmer, Natalie Schluter, Nathan Schneider, and Xiaodan Zhu

# Organizing Committee

## SemEval Organizers:

Alexis Palmer, University of Colorado Boulder  
Nathan Schneider, Georgetown University  
Natalie Schluter, IT University of Copenhagen & Google Brain  
Guy Emerson, University of Cambridge  
Xiaodan Zhu, Queen's University  
Aurelie Herbelot, University of Trento

## Assistant Organizers:

Julia R. Bonn, University of Colorado Boulder  
Abhidip Bhattacharyya, University of Colorado Boulder

## Task Selection Committee:

Eneko Agirre, Francesco Barbieri, Alberto Barrón-Cedeño, Valerio Basile, Steven Bethard, Georgeta Bordea, Davide Buscaldi, Tanmoy Chakraborty, Wanxiang Che, Colin Cherry, Keith Cortis, Giovanni Da San Martino, Amitava Das, Tobias Daudert, Franck Dernoncourt, Luis Espinosa Anke, André Freitas, Dimitris Galanis, Michael Gamon, Goran Glavaš, Ivan Habernal, Simon Hengchen, Nabil Hossain, Filip Ilievski, Els Lefever, Alessandro Lenci, Shuailong Liang, Nedim Lipka, Shervin Malmasi, Jonathan May, Tristan Miller, Ashutosh Modi, Saif Mohammad, Preslav Nakov, Roberto Navigli, Mohammad Taher Pilehvar, Manfred Pinkal, Maria Pontiki, Soujanya Poria, Marten Postma, Matthew Purver, Alan Ritter, Marko Robnik-Sikonja, Francesco Ronzano, Michael Roth, Mohammad Salameh, Juliano Efsen Sales, Dominik Schlechtweg, Fabrizio Sebastiani, Ekaterina Shutova, Vered Shwartz, Sasha Spala, Karin Verspoor, Cunxiang Wang, Xiaoyu Yang, Marcos Zampieri, Qiong Zhang, Arkaitz Zubiaga

## Task Organizers:

Task 1: Matthew Shardlow, Richard Evans, Gustavo Henrique Paetzold, Marcos Zampieri  
Task 2: Federico Martelli, Najla Kalach, Gabriele Tola, Roberto Navigli  
Task 4: Boyuan Zheng, Xiaoyu Yang, Yu-Ping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, Xiaodan Zhu  
Task 5: John Pavlopoulos, Ion Androutsopoulos, Jeffrey Sorensen, Léo Laugier  
Task 6: Giovanni Da San Martino, Hamed Firooz, Preslav Nakov, Fabrizio Silvestri  
Task 7: J. A. Meaney, Steven Wilson, Walid Magdy, Luis Chiruzzo  
Task 8: Corey Harper, Jessica Cox, Ron Daniel, Paul Groth, Curt Kohler, Antony Scerri  
Task 9: Nancy Xin Ru Wang, Sara Rosenthal, Marina Danilevsky, Diwakar Mahajan  
Task 10: Steven Bethard, Egoitz Laparra, Timothy Miller, Özlem Uzuner  
Task 11: Jennifer D'Souza, Sören Auer, Ted Pedersen  
Task 12: Alexandra Uma, Tommaso Fornaciari, Tristan Miller, Barbara Plank, Edwin Simpson, Massimo Poesio

## Invited Speakers:

Diyi Yang, Georgia Institute of Technology (shared speaker with \*SEM)  
Hannah Rohde, University of Edinburgh



## Table of Contents

<i>SemEval-2021 Task 1: Lexical Complexity Prediction</i>	
Matthew Shardlow, Richard Evans, Gustavo Henrique Paetzold and Marcos Zampieri . . . . .	1
<i>OCHADAI-KYOTO at SemEval-2021 Task 1: Enhancing Model Generalization and Robustness for Lexical Complexity Prediction</i>	
Yuki Taya, Lis Kanashiro Pereira, Fei Cheng and Ichiro Kobayashi . . . . .	17
<i>SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC)</i>	
Federico Martelli, Najla Kalach, Gabriele Tola and Roberto Navigli . . . . .	24
<i>SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning</i>	
Boyuan Zheng, Xiaoyu Yang, Yu-Ping Ruan, Zhenhua Ling, Quan Liu, Si Wei and Xiaodan Zhu	37
<i>TA-MAMC at SemEval-2021 Task 4: Task-adaptive Pretraining and Multi-head Attention for Abstract Meaning Reading Comprehension</i>	
Jing Zhang, Yimeng Zhuang and Yinpei Su . . . . .	51
<i>SemEval-2021 Task 5: Toxic Spans Detection</i>	
John Pavlopoulos, Jeffrey Sorensen, Léo Laugier and Ion Androutsopoulos . . . . .	59
<i>SemEval-2021 Task 6: Detection of Persuasion Techniques in Texts and Images</i>	
Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov and Giovanni Da San Martino . . . . .	70
<i>Alpha at SemEval-2021 Task 6: Transformer Based Propaganda Classification</i>	
Zhida Feng, Jiji Tang, Jiaxiang Liu, Weichong Yin, Shikun Feng, Yu Sun and Li Chen . . . . .	99
<i>SemEval 2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense</i>	
J. A. Meaney, Steven Wilson, Luis Chiruzzo, Adam Lopez and Walid Magdy . . . . .	105
<i>LangResearchLab NC at SemEval-2021 Task 1: Linguistic Feature Based Modelling for Lexical Complexity</i>	
Raksha Agarwal and Niladri Chatterjee . . . . .	120
<i>Complex words identification using word-level features for SemEval-2020 Task 1</i>	
Jenny A. Ortiz-Zambrano and Arturo Montejo-Ráez . . . . .	126
<i>TUDA-CCL at SemEval-2021 Task 1: Using Gradient-boosted Regression Tree Ensembles Trained on a Heterogeneous Feature Set for Predicting Lexical Complexity</i>	
Sebastian Gombert and Sabine Bartsch . . . . .	130
<i>JCT at SemEval-2021 Task 1: Context-aware Representation for Lexical Complexity Prediction</i>	
Chaya Liebeskind, Otniel Elkayam and Shmuel Liebeskind . . . . .	138
<i>IAPUCP at SemEval-2021 Task 1: Stacking Fine-Tuned Transformers is Almost All You Need for Lexical Complexity Prediction</i>	
Kervy Rivas Rojas and Fernando Alva-Manchego . . . . .	144
<i>Uppsala NLP at SemEval-2021 Task 2: Multilingual Language Models for Fine-tuning and Feature Extraction in Word-in-Context Disambiguation</i>	
Huiling You, Xingran Zhu and Sara Stymne . . . . .	150

<i>SkoltechNLP at SemEval-2021 Task 2: Generating Cross-Lingual Training Data for the Word-in-Context Task</i>	
Anton Razzhigaev, Nikolay Arefyev and Alexander Panchenko . . . . .	157
<i>Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning</i>	
Boris Zhestiankin and Maria Ponomareva . . . . .	163
<i>SzegedAI at SemEval-2021 Task 2: Zero-shot Approach for Multilingual and Cross-lingual Word-in-Context Disambiguation</i>	
Gábor Berend . . . . .	169
<i>ReCAM@IITK at SemEval-2021 Task 4: BERT and ALBERT based Ensemble for Abstract Word Prediction</i>	
Abhishek Mittal and Ashutosh Modi . . . . .	175
<i>ECNU_JCA_1 SemEval-2021 Task 4: Leveraging Knowledge-enhanced Graph Attention Networks for Reading Comprehension of Abstract Meaning</i>	
Pingsheng Liu, Linlin Wang, Qian Zhao, Hao Chen, Yuxi Feng, Xin Lin and liang he . . . . .	183
<i>LRG at SemEval-2021 Task 4: Improving Reading Comprehension with Abstract Words using Augmentation, Linguistic Features and Voting</i>	
Abheesht Sharma, Harshit Pandey, Gunjan Chhablani, Yash Bhartia and Tirtharaj Dash . . . . .	189
<i>IIE-NLP-Eyas at SemEval-2021 Task 4: Enhancing PLM for ReCAM with Special Tokens, Re-Ranking, Siamese Encoders and Back Translation</i>	
Yuqiang Xie, Luxi Xing, Wei Peng and Yue Hu . . . . .	199
<i>NLP-IIS@UT at SemEval-2021 Task 4: Machine Reading Comprehension using the Long Document Transformer</i>	
Hossein Basafa, Sajad Movahedi, Ali Ebrahimi, Azadeh Shakery and Hesham Faili . . . . .	205
<i>IITK@Detox at SemEval-2021 Task 5: Semi-Supervised Learning and Dice Loss for Toxic Spans Detection</i>	
Archit Bansal, Abhay Kaushik and Ashutosh Modi . . . . .	211
<i>UniParma at SemEval-2021 Task 5: Toxic Spans Detection Using CharacterBERT and Bag-of-Words Model</i>	
Akbar Karimi, Leonardo Rossi and Andrea Prati . . . . .	220
<i>UPB at SemEval-2021 Task 5: Virtual Adversarial Training for Toxic Spans Detection</i>	
Andrei Paraschiv, Dumitru-Clementin Cercel and Mihai Dascalu . . . . .	225
<i>NLRG at SemEval-2021 Task 5: Toxic Spans Detection Leveraging BERT-based Token Classification and Span Prediction Techniques</i>	
Gunjan Chhablani, Abheesht Sharma, Harshit Pandey, Yash Bhartia and Shan Suthaharan . . . . .	233
<i>UoB at SemEval-2021 Task 5: Extending Pre-Trained Language Models to Include Task and Domain-Specific Information for Toxic Span Prediction</i>	
Erik Yan and Harish Tayyar Madabushi . . . . .	243
<i>Cisco at SemEval-2021 Task 5: What's Toxic?: Leveraging Transformers for Multiple Toxic Span Extraction from Online Comments</i>	
Sreyan Ghosh and Sonal Kumar . . . . .	249



<i>MedAI at SemEval-2021 Task 5: Start-to-end Tagging Framework for Toxic Spans Detection</i>	
Zhen Wang, Hongjie Fan and Junfei Liu .....	258
<i>HamiltonDinggg at SemEval-2021 Task 5: Investigating Toxic Span Detection using RoBERTa Pre-training</i>	
Huiyang Ding and David Jurgens.....	263
<i>WVOQ at SemEval-2021 Task 6: BART for Span Detection and Classification</i>	
Cees Roele.....	270
<i>HumorHunter at SemEval-2021 Task 7: Humor and Offense Recognition with Disentangled Attention</i>	
Yubo Xie, Junze Li and Pearl Pu .....	275
<i>Grenzlinie at SemEval-2021 Task 7: Detecting and Rating Humor and Offense</i>	
Renyuan Liu and Xiaobing Zhou .....	281
<i>abcbpc at SemEval-2021 Task 7: ERNIE-based Multi-task Model for Detecting and Rating Humor and Offense</i>	
Chao Pang, Xiaoran Fan, Weiyue Su, Xuyi Chen, Shuohuan Wang, Jiayang Liu, Xuan Ouyang, Shikun Feng and Yu Sun .....	286
<i>Humor@IITK at SemEval-2021 Task 7: Large Language Models for Quantifying Humor and Offensiveness</i>	
Aishwarya Gupta, Avik Pal, Bholeshwar Khurana, Lakshay Tyagi and Ashutosh Modi.....	290
<i>RoMa at SemEval-2021 Task 7: A Transformer-based Approach for Detecting and Rating Humor and Offense</i>	
Roberto Labadie, Mariano Jason Rodriguez, Reynier Ortega and Paolo Rosso .....	297
<i>SemEval-2021 Task 8: MeasEval – Extracting Counts and Measurements and their Related Contexts</i>	
Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr. and Paul Groth .....	306
<i>SemEval-2021 Task 9: Fact Verification and Evidence Finding for Tabular Data in Scientific Documents (SEM-TAB-FACTS)</i>	
Nancy X. R. Wang, Diwakar Mahajan, Marina Danilevsky and Sara Rosenthal.....	317
<i>BreakingBERT@IITK at SemEval-2021 Task 9: Statement Verification and Evidence Finding with Tables</i>	
Aditya Jindal, Ankur Gupta, Jaya Srivastava, Preeti Menghwani, Vijit Malik, Vishesh Kaushik and Ashutosh Modi .....	327
<i>SemEval-2021 Task 12: Learning with Disagreements</i>	
Alexandra Uma, Tommaso Fornaciari, Anca Dumitrache, Tristan Miller, Jon Chamberlain, Barbara Plank, Edwin Simpson and Massimo Poesio .....	338
<i>SemEval-2021 Task 10: Source-Free Domain Adaptation for Semantic Processing</i>	
Egoitz Laparra, Xin Su, Yiyun Zhao, Özlem Uzuner, Timothy Miller and Steven Bethard .....	348
<i>BLCUFIGHT at SemEval-2021 Task 10: Novel Unsupervised Frameworks For Source-Free Domain Adaptation</i>	
Weikang Wang, Yi Wu, Yixiang Liu and Pengyuan Liu .....	357
<i>SemEval-2021 Task 11: NLPContributionGraph - Structuring Scholarly NLP Contributions for a Research Knowledge Graph</i>	
Jennifer D’Souza, Sören Auer and Ted Pedersen .....	364

<i>UIUC_BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions</i>	
Haoyang Liu, M. Janina Sarol and Halil Kilicoglu .....	377
<i>KGP at SemEval-2021 Task 8: Leveraging Multi-Staged Language Models for Extracting Measurements, their Attributes and Relations</i>	
Neel Karia, Ayush Kaushal and Faraaz Mallick .....	387
<i>DPR at SemEval-2021 Task 8: Dynamic Path Reasoning for Measurement Relation Extraction</i>	
Amir Pouran Ben Veyseh, Franck Dernoncourt and Thien Huu Nguyen .....	397
<i>CLaC-np at SemEval-2021 Task 8: Dependency DGCNN</i>	
Nihatha Lathiff, Pavel PK Khloponin and Sabine Bergler .....	404
<i>CLaC-BP at SemEval-2021 Task 8: SciBERT Plus Rules for MeasEval</i>	
Benjamin Therien, Parsa Bagherzadeh and Sabine Bergler .....	410
<i>THiFly_Queens at SemEval-2021 Task 9: Two-stage Statement Verification with Adaptive Ensembling and Slot-based Operation</i>	
Yuxuan Zhou, Kaiyin Zhou, Xien Liu, Ji Wu and Xiaodan Zhu .....	416
<i>TAPAS at SemEval-2021 Task 9: Reasoning over tables with intermediate pre-training</i>	
Thomas Müller, Julian Eisenschlos and Syrine Krichene .....	423
<i>BOUN at SemEval-2021 Task 9: Text Augmentation Techniques for Fact Verification in Tabular Data</i>	
Abdullatif Köksal, Yusuf Yüksel, Bekir Yıldırım and Arzucan Özgür .....	431
<i>IITK at SemEval-2021 Task 10: Source-Free Unsupervised Domain Adaptation using Class Prototypes</i>	
Harshit Kumar, Jinang Shah, Nidhi Hegde, Priyanshu Gupta, Vaibhav Jindal and Ashutosh Modi	438
<i>PTST-UoM at SemEval-2021 Task 10: Parsimonious Transfer for Sequence Tagging</i>	
Kemal Kurniawan, Lea Frermann, Philip Schulz and Trevor Cohn .....	445
<i>Self-Adapter at SemEval-2021 Task 10: Entropy-based Pseudo-Labeler for Source-free Domain Adaptation</i>	
Sangwon Yoon, Yanghoon Kim and Kyomin Jung .....	452
<i>The University of Arizona at SemEval-2021 Task 10: Applying Self-training, Active Learning and Data Augmentation to Source-free Domain Adaptation</i>	
Xin Su, Yiyun Zhao and Steven Bethard .....	458
<i>KnowGraph@IITK at SemEval-2021 Task 11: Building Knowledge Graph for NLP Research</i>	
Shashank Shailabh, Sajal Chaurasia and Ashutosh Modi .....	467
<i>YNU-HPCC at SemEval-2021 Task 11: Using a BERT Model to Extract Contributions from NLP Scholarly Articles</i>	
Xinge Ma, Jin Wang and Xuejie Zhang .....	478
<i>ITNLP at SemEval-2021 Task 11: Boosting BERT with Sampling and Adversarial Training for Knowledge Extraction</i>	
Genyu Zhang, Yu Su, Changhong He, Lei Lin, Chengjie Sun and Lili Shan .....	485

<i>Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction</i>	
Anna Martin and Ted Pedersen . . . . .	490
<i>INNOVATORS at SemEval-2021 Task-11: A Dependency Parsing and BERT-based model for Extracting Contribution Knowledge from Scientific Papers</i>	
Hardik Arora, Tirthankar Ghosal, Sandeep Kumar, Suraj Patwal and Phil Gooch . . . . .	502
<i>MCL@IITK at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation using Augmented Data, Signals, and Transformers</i>	
Rohan Gupta, Jay Mundra, Deepak Mahajan and Ashutosh Modi . . . . .	511
<i>HITSZ-HLT at SemEval-2021 Task 5: Ensemble Sequence Labeling and Span Boundary Detection for Toxic Span Detection</i>	
Qinglin Zhu, Zijie Lin, Yice Zhang, Jingyi Sun, Xiang Li, Qihui Lin, Yixue Dang and Ruifeng Xu	521
<i>SarcasmDet at SemEval-2021 Task 7: Detect Humor and Offensive based on Demographic Factors using RoBERTa Pre-trained Model</i>	
Dalya Faraj and Malak Abdullah . . . . .	527
<i>UPB at SemEval-2021 Task 8: Extracting Semantic Information on Measurements as Multi-Turn Question Answering</i>	
Andrei-Marius Avram, George-Eduard Zaharia, Dumitru-Clementin Cercel and Mihai Dascalu	534
<i>IITK@LCP at SemEval-2021 Task 1: Classification for Lexical Complexity Regression Task</i>	
Neil Shirude, Sagnik Mukherjee, Tushar Shandhilya, Ananta Mukherjee and Ashutosh Modi . .	541
<i>LCP-RIT at SemEval-2021 Task 1: Exploring Linguistic Features for Lexical Complexity Prediction</i>	
Abhinandan Tejalkumar Desai, Kai North, Marcos Zampieri and Christopher Homan . . . . .	548
<i>Alejandro Mosquera at SemEval-2021 Task 1: Exploring Sentence and Word Features for Lexical Complexity Prediction</i>	
Alejandro Mosquera . . . . .	554
<i>CompNA at SemEval-2021 Task 1: Prediction of lexical complexity analyzing heterogeneous features</i>	
Giuseppe Vettigli and Antonio Sorgente . . . . .	560
<i>PolyU CBS-Comp at SemEval-2021 Task 1: Lexical Complexity Prediction (LCP)</i>	
Rong Xiang, Jinghang Gu, Emmanuele Chersoni, Wenjie Li, Qin Lu and Chu-Ren Huang . . . .	565
<i>LAST at SemEval-2021 Task 1: Improving Multi-Word Complexity Prediction Using Bigram Association Measures</i>	
Yves Bestgen . . . . .	571
<i>DeepBlueAI at SemEval-2021 Task 1: Lexical Complexity Prediction with A Deep Ensemble Approach</i>	
Chunguang Pan, Bingyan Song, Shengguang Wang and Zhipeng Luo . . . . .	578
<i>CS-UM6P at SemEval-2021 Task 1: A Deep Learning Model-based Pre-trained Transformer Encoder for Lexical Complexity</i>	
Nabil El Mamoun, Abdelkader El Mahdaouy, Abdellah El Mekki, Kabil Essefar and Ismail Berrada	585

<i>Cambridge at SemEval-2021 Task 1: An Ensemble of Feature-Based and Neural Models for Lexical Complexity Prediction</i>	
Zheng Yuan, Gladys Tyen and David Strohmaier . . . . .	590
<i>hub at SemEval-2021 Task 1: Fusion of Sentence and Word Frequency to Predict Lexical Complexity</i>	
Bo Huang, Yang Bai and Xiaobing Zhou . . . . .	598
<i>Manchester Metropolitan at SemEval-2021 Task 1: Convolutional Networks for Complex Word Identification</i>	
Robert Flynn and Matthew Shardlow . . . . .	603
<i>UPB at SemEval-2021 Task 1: Combining Deep Learning and Hand-Crafted Features for Lexical Complexity Prediction</i>	
George-Eduard Zaharia, Dumitru-Clementin Cercel and Mihai Dascalu . . . . .	609
<i>UTFPR at SemEval-2021 Task 1: Complexity Prediction by Combining BERT Vectors and Classic Features</i>	
Gustavo Henrique Paetzold . . . . .	617
<i>RG PA at SemEval-2021 Task 1: A Contextual Attention-based Model with RoBERTa for Lexical Complexity Prediction</i>	
Gang Rao, Maochang Li, Xiaolong Hou, Lianxin Jiang, Yang Mo and Jianping Shen . . . . .	623
<i>CSECU-DSG at SemEval-2021 Task 1: Fusion of Transformer Models for Lexical Complexity Prediction</i>	
Abdul Aziz, MD. Akram Hossain and Abu Nowshed Chy . . . . .	627
<i>CLULEX at SemEval-2021 Task 1: A Simple System Goes a Long Way</i>	
Greta Smolenska, Peter Kolb, Sinan Tang, Mironas Bitinis, Héctor Hernández and Elin Asklöv	632
<i>RS_GV at SemEval-2021 Task 1: Sense Relative Lexical Complexity Prediction</i>	
Regina Stodden and Gayatri Venugopal . . . . .	640
<i>UNBNLP at SemEval-2021 Task 1: Predicting lexical complexity with masked language models and character-level encoders</i>	
Milton King, Ali Hakimi Parizi, Samin Fakharian and Paul Cook . . . . .	650
<i>ANDI at SemEval-2021 Task 1: Predicting complexity in context using distributional models, behavioural norms, and lexical resources</i>	
Armand Rotaru . . . . .	655
<i>JUST-BLUE at SemEval-2021 Task 1: Predicting Lexical Complexity using BERT and RoBERTa Pre-trained Language Models</i>	
Tuqa Bani Yaseen, Qusai Ismail, Sarah Al-Omari, Eslam Al-Sobh and Malak Abdullah . . . . .	661
<i>BigGreen at SemEval-2021 Task 1: Lexical Complexity Prediction with Assembly Models</i>	
Aadil Islam, Weicheng Ma and Soroush Vosoughi . . . . .	667
<i>cs60075_team2 at SemEval-2021 Task 1 : Lexical Complexity Prediction using Transformer-based Language Models pre-trained on various text corpora</i>	
Abhilash Nandy, Sayantan Adak, Tanurima Halder and Sai Mahesh Pokala . . . . .	678
<i>C3SL at SemEval-2021 Task 1: Predicting Lexical Complexity of Words in Specific Contexts with Sentence Embeddings</i>	
Raul Almeida, Hegler Tissot and Marcos Didonet Del Fabro . . . . .	683

<i>Stanford MLab at SemEval-2021 Task 1: Tree-Based Modelling of Lexical Complexity using Word Embeddings</i>	
Erik Rozi, Niveditha Iyer, Gordon Chi, Enok Choe, Kathy J. Lee, Kevin Liu, Patrick Liu, Zander Lack, Jillian Tang and Ethan A. Chi .....	688
<i>archer at SemEval-2021 Task 1: Contextualising Lexical Complexity</i>	
Irene Russo .....	694
<i>katildakat at SemEval-2021 Task 1: Lexical Complexity Prediction of Single Words and Multi-Word Expressions in English</i>	
Katja Voskoboynik .....	700
<i>GX at SemEval-2021 Task 2: BERT with Lemma Information for MCL-WiC Task</i>	
Wanying Xie .....	706
<i>PALI at SemEval-2021 Task 2: Fine-Tune XLM-RoBERTa for Word in Context Disambiguation</i>	
Shuyi Xie, Jian Ma, Haiqin Yang, Lianxin Jiang, Yang Mo and Jianping Shen .....	713
<i>hub at SemEval-2021 Task 2: Word Meaning Similarity Prediction Model Based on RoBERTa and Word Frequency</i>	
Bo Huang, Yang Bai and Xiaobing Zhou .....	719
<i>Lotus at SemEval-2021 Task 2: Combination of BERT and Paraphrasing for English Word Sense Disambiguation</i>	
Niloofer Ranjbar and Hossein Zeinali .....	724
<i>Cambridge at SemEval-2021 Task 2: Neural WiC-Model with Data Augmentation and Exploration of Representation</i>	
Zheng Yuan and David Strohmaier .....	730
<i>UoB_UK at SemEval 2021 Task 2: Zero-Shot and Few-Shot Learning for Multi-lingual and Cross-lingual Word Sense Disambiguation.</i>	
Wei Li, Harish Tayyar Madabushi and Mark Lee .....	738
<i>PAW at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation : Exploring Cross Lingual Transfer, Augmentations and Adversarial Training</i>	
Harsh Goyal, Aadarsh Singh and Priyanshu Kumar .....	743
<i>LU-BZU at SemEval-2021 Task 2: Word2Vec and Lemma2Vec performance in Arabic Word-in-Context disambiguation</i>	
Moustafa Al-Hajj and Mustafa Jarrar .....	748
<i>GlossReader at SemEval-2021 Task 2: Reading Definitions Improves Contextualized Word Embeddings</i>	
Maxim Rachinskiy and Nikolay Arefyev .....	756
<i>UALberta at SemEval-2021 Task 2: Determining Sense Synonymy via Translations</i>	
Bradley Hauer, Hongchang Bao, Arnob Mallik and Grzegorz Kondrak .....	763
<i>TransWiC at SemEval-2021 Task 2: Transformer-based Multilingual and Cross-lingual Word-in-Context Disambiguation</i>	
Hansi Hettiarachchi and Tharindu Ranasinghe .....	771
<i>LIORI at SemEval-2021 Task 2: Span Prediction and Binary Classification approaches to Word-in-Context Disambiguation</i>	
Adis Davletov, Nikolay Arefyev, Denis Gordeev and Alexey Rey .....	780

<i>FII_CROSS at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation</i> Ciprian Bodnar, Andrada Tapuc, Cosmin Pintilie, Daniela Gifu and Diana Trandabat . . . . .	787
<i>XRJL-HKUST at SemEval-2021 Task 4: WordNet-Enhanced Dual Multi-head Co-Attention for Reading Comprehension of Abstract Meaning</i> Yuxin Jiang, Ziyi Shou, Qijun Wang, Hao Wu and Fangzhen Lin . . . . .	793
<i>UoR at SemEval-2021 Task 4: Using Pre-trained BERT Token Embeddings for Question Answering of Abstract Meaning</i> Thanet Markchom and Huizhi Liang . . . . .	799
<i>Noobs at Semeval-2021 Task 4: Masked Language Modeling for abstract answer prediction</i> Shikhar Shukla, Sarthak Sarthak and Karm Veer Arya . . . . .	805
<i>ZJUKLAB at SemEval-2021 Task 4: Negative Augmentation with Language Model for Reading Comprehension of Abstract Meaning</i> Xin Xie, Xiangnan Chen, Xiang Chen, Yong Wang, Ningyu Zhang, Shumin Deng and Huajun Chen . . . . .	810
<i>PINGAN Omini-Sinitic at SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning</i> Ye Wang, Yanmeng Wang, Haijun Zhu, Bo Zeng, Zhenghong Hao, Shaojun Wang and Jing Xiao . . . . .	820
<i>NEUer at SemEval-2021 Task 4: Complete Summary Representation by Filling Answers into Question for Matching Reading Comprehension</i> Zhixiang Chen, yikun lei, Pai Liu and Guibing Guo . . . . .	827
<i>WLV-RIT at SemEval-2021 Task 5: A Neural Transformer Framework for Detecting Toxic Spans</i> Tharindu Ranasinghe, Diptanu Sarkar, Marcos Zampieri and Alexander Ororbia . . . . .	833
<i>YNU-HPCC at SemEval-2021 Task 5: Using a Transformer-based Model with Auxiliary Information for Toxic Span Detection</i> Ruijun Chen, Jin Wang and Xuejie Zhang . . . . .	841
<i>UIT-ISE-NLP at SemEval-2021 Task 5: Toxic Spans Detection with BiLSTM-CRF and ToxicBERT Comment Classification</i> Son T. Luu and Ngan Nguyen . . . . .	846
<i>GHOST at SemEval-2021 Task 5: Is explanation all you need?</i> Kamil Pluciński and Hanna Klimczak . . . . .	852
<i>GoldenWind at SemEval-2021 Task 5: Orthrus - An Ensemble Approach to Identify Toxicity</i> Marco Palomino, Dawid Grad and James Bedwell . . . . .	860
<i>LISAC FSDM USMBA at SemEval-2021 Task 5: Tackling Toxic Spans Detection Challenge with Supervised SpanBERT-based Model and Unsupervised LIME-based Model</i> Abdessamad Benlahbib, Ahmed Alami and Hamza Alami . . . . .	865
<i>HITMI&amp;T at SemEval-2021 Task 5: Integrating Transformer and CRF for Toxic Spans Detection</i> Chenyi Wang, Tianshu Liu and Tiejun Zhao . . . . .	870
<i>AStarTwice at SemEval-2021 Task 5: Toxic Span Detection Using RoBERTa-CRF, Domain Specific Pre-Training and Self-Training</i> Thakur Ashutosh Suman and Abhinav Jain . . . . .	875

<i>NLP_UIOWA at Semeval-2021 Task 5: Transferring Toxic Sets to Tag Toxic Spans</i> Jonathan Rusert .....	881
<i>S-NLP at SemEval-2021 Task 5: An Analysis of Dual Networks for Sequence Tagging</i> Viet Anh Nguyen, Tam Minh Nguyen, Huy Quang Dao and Quang Huu Pham .....	888
<i>UAntwerp at SemEval-2021 Task 5: Spans are Spans, stacking a binary word level approach to toxic span detection</i> Ben Burtenshaw and Mike Kestemont .....	898
<i>hub at SemEval-2021 Task 5: Toxic Span Detection Based on Word-Level Classification</i> Bo Huang, Yang Bai and Xiaobing Zhou .....	904
<i>Sefamerve ARGE at SemEval-2021 Task 5: Toxic Spans Detection Using Segmentation Based 1-D Convolutional Neural Network Model</i> Selman Delil, Birol Kuyumcu and Cüneyt Aksakallı .....	909
<i>MIPT-NSU-UTMN at SemEval-2021 Task 5: Ensembling Learning with Pre-trained Language Models for Toxic Spans Detection</i> Mikhail Kotyushev, Anna Glazkova and Dmitry Morozov .....	913
<i>UIT-E10dot3 at SemEval-2021 Task 5: Toxic Spans Detection with Named Entity Recognition and Question-Answering Approaches</i> Phu Gia Hoang, Luan Thanh Nguyen and Kiet Nguyen .....	919
<i>SkoltechNLP at SemEval-2021 Task 5: Leveraging Sentence-level Pre-training for Toxic Span Detection</i> David Dale, Igor Markov, Varvara Logacheva, Olga Kozlova, Nikita Semenov and Alexander Panchenko .....	927
<i>Entity at SemEval-2021 Task 5: Weakly Supervised Token Labelling for Toxic Spans Detection</i> Vaibhav Jain and Mina Naghshnejad .....	935
<i>BennettNLP at SemEval-2021 Task 5: Toxic Spans Detection using Stacked Embedding Powered Toxic Entity Recognizer</i> Harsh Kataria, Ambuje Gupta and Vipul Mishra .....	941
<i>UoT-UWF-PartAI at SemEval-2021 Task 5: Self Attention Based Bi-GRU with Multi-Embedding Representation for Toxicity Highlighter</i> Hamed Babaei Giglou, Taher Rahgooy, Mostafa Rahgouy and Jafar Razmara .....	948
<i>YoungSheldon at SemEval-2021 Task 5: Fine-tuning Pre-trained Language Models for Toxic Spans Detection using Token classification Objective</i> Mayukh Sharma, Ilanthenral Kandasamy and W.B. Vasantha .....	953
<i>HLE-UPC at SemEval-2021 Task 5: Multi-Depth DistilBERT for Toxic Spans Detection</i> Rafel Palliser-Sans and Albert Rial-Farràs .....	960
<i>Lone Pine at SemEval-2021 Task 5: Fine-Grained Detection of Hate Speech Using BERToxic</i> Yakoob Khan, Weicheng Ma and Soroush Vosoughi .....	967
<i>SRPOL DIALOGUE SYSTEMS at SemEval-2021 Task 5: Automatic Generation of Training Data for Toxic Spans Detection</i> Michał Satława, Katarzyna Zamłyńska, Jarosław Piersa, Joanna Kolis, Klaudia Firląg, Katarzyna Beksa, Zuzanna Bordzicka, Christian Goltz, Paweł Bujnowski and Piotr Andruszkiewicz .....	974

<i>SINAI at SemEval-2021 Task 5: Combining Embeddings in a BiLSTM-CRF model for Toxic Spans Detection</i>	
Flor Miriam Plaza-del-Arco, Pilar López-Úbeda, L. Alfonso Ureña-López and M. Teresa Martín-Valdivia .....	984
<i>CSECU-DSG at SemEval-2021 Task 5: Leveraging Ensemble of Sequence Tagging Models for Toxic Spans Detection</i>	
Tashin Hossain, Jannatun Naim, Fareen Tasneem, Radiathun Tasnia and Abu Nowshed Chy ...	990
<i>UTNLP at SemEval-2021 Task 5: A Comparative Analysis of Toxic Span Detection using Attention-based, Named Entity Recognition, and Ensemble Models</i>	
Alireza Salemi, Nazanin Sabri, Emad Kebriaei, Behnam Bahrak and Azadeh Shakery .....	995
<i>macech at SemEval-2021 Task 5: Toxic Spans Detection</i>	
Maggie Cech .....	1003
<i>LZ1904 at SemEval-2021 Task 5: Bi-LSTM-CRF for Toxic Span Detection using Pretrained Word Embedding</i>	
Liang Zou and Wen Li .....	1009
<i>LIIR at SemEval-2021 task 6: Detection of Persuasion Techniques In Texts and Images using CLIP features</i>	
Erfan Ghadery, Damien Sileo and Marie-Francine Moens .....	1015
<i>AIMH at SemEval-2021 Task 6: Multimodal Classification Using an Ensemble of Transformer Models</i>	
Nicola Messina, Fabrizio Falchi, Claudio Gennaro and Giuseppe Amato .....	1020
<i>HOMADOS at SemEval-2021 Task 6: Multi-Task Learning for Propaganda Detection</i>	
Konrad Kaczyński and Piotr Przybyła .....	1027
<i>1213Li at SemEval-2021 Task 6: Detection of Propaganda with Multi-modal Attention and Pre-trained Models</i>	
Peiguang Li, Xuan Li and Xian Sun .....	1032
<i>NlyticsFKIE at SemEval-2021 Task 6: Detection of Persuasion Techniques In Texts And Images</i>	
Albert Pritzkau .....	1037
<i>YNU-HPCC at SemEval-2021 Task 6: Combining ALBERT and Text-CNN for Persuasion Detection in Texts and Images</i>	
Xingyu Zhu, Jin Wang and Xuejie Zhang .....	1045
<i>LT3 at SemEval-2021 Task 6: Using Multi-Modal Compact Bilinear Pooling to Combine Visual and Textual Understanding in Memes</i>	
Pranaydeep Singh and Els Lefever .....	1051
<i>FPAI at SemEval-2021 Task 6: BERT-MRC for Propaganda Techniques Detection</i>	
Xiaolong Hou, Junsong Ren, Gang Rao, Lianxin Lian, Zhihao Ruan, Yang Mo and Jianping Shen	1056
<i>NLPITR at SemEval-2021 Task 6: RoBERTa Model with Data Augmentation for Persuasion Techniques Detection</i>	
Vansh Gupta and Raksha Sharma .....	1061



<i>LeCun at SemEval-2021 Task 6: Detecting Persuasion Techniques in Text Using Ensembled Pretrained Transformers and Data Augmentation</i>	
Dia Abujaber, Ahmed Qarqaz and Malak A. Abdullah . . . . .	1068
<i>Volta at SemEval-2021 Task 6: Towards Detecting Persuasive Texts and Images using Textual and Multimodal Ensemble</i>	
Kshitij Gupta, Devansh Gautam and Radhika Mamidi . . . . .	1075
<i>MinD at SemEval-2021 Task 6: Propaganda Detection using Transfer Learning and Multimodal Fusion</i>	
Junfeng Tian, Min Gui, Chenliang Li, Ming Yan and Wenming Xiao . . . . .	1082
<i>CSECU-DSG at SemEval-2021 Task 6: Orchestrating Multimodal Neural Architectures for Identifying Persuasion Techniques in Texts and Images</i>	
Tashin Hossain, Jannatun Naim, Fareen Tasneem, Radiathun Tasnia and Abu Nowshed Chy .	1088
<i>UMUTeam at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with Linguistic Features and Word Embeddings</i>	
José Antonio García-Díaz and Rafael Valencia-García . . . . .	1096
<i>ES-JUST at SemEval-2021 Task 7: Detecting and Rating Humor and Offensive Text Using Deep Learning</i>	
Emran Al Bashabsheh and Sanaa Abu Alasal . . . . .	1102
<i>Tsia at SemEval-2021 Task 7: Detecting and Rating Humor and Offense</i>	
Zhengyi Guan and Xiaobing ZXB Zhou . . . . .	1108
<i>DLJUST at SemEval-2021 Task 7: Hahackathon: Linking Humor and Offense</i>	
Hani Al-Omari, Isra'a AbedulNabi and Rehab Duwairi . . . . .	1114
<i>Gulu at SemEval-2021 Task 7: Detecting and Rating Humor and Offense</i>	
Maoqin Yang . . . . .	1120
<i>DUTH at SemEval-2021 Task 7: Is Conventional Machine Learning for Humorous and Offensive Tasks enough in 2021?</i>	
Alexandros Karasakalidis, Dimitrios Effrosynidis and Avi Arampatzis . . . . .	1125
<i>DeepBlueAI at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with Stacking Diverse Language Model-Based Methods</i>	
Bingyan Song, Chunguang Pan, Shengguang Wang and Zhipeng Luo . . . . .	1130
<i>CS-UM6P at SemEval-2021 Task 7: Deep Multi-Task Learning Model for Detecting and Rating Humor and Offense</i>	
Kabil Essefar, Abdellah El Mekki, Abdelkader El Mahdaouy, Nabil El Mamoun and Ismail Berrada	1135
<i>hub at SemEval-2021 Task 7: Fusion of ALBERT and Word Frequency Information Detecting and Rating Humor and Offense</i>	
Bo Huang and Yang Bai . . . . .	1141
<i>YoungSheldon at SemEval-2021 Task 7: Fine-tuning Is All You Need</i>	
Mayukh Sharma, Ilanthenral Kandasamy and W.B. Vasantha . . . . .	1146
<i>MagicPai at SemEval-2021 Task 7: Method for Detecting and Rating Humor Based on Multi-Task Adversarial Training</i>	
Jian Ma, Shuyi Xie, Haiqin Yang, Lianxin Jiang, Mengyuan Zhou, Xiaoyi Ruan and Yang Mo	1153

<i>UPB at SemEval-2021 Task 7: Adversarial Multi-Task Learning for Detecting and Rating Humor and Offense</i>	
Răzvan-Alexandru Smădu, Dumitru-Clementin Cercel and Mihai Dascalu	1160
<i>Team_KGP at SemEval-2021 Task 7: A Deep Neural System to Detect Humor and Offense with Their Ratings in the Text Data</i>	
Anik Mondal and Raksha Sharma	1169
<i>ZYJ at SemEval-2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense with ALBERT-Based Model</i>	
Yingjia Zhao and Xin Tao	1175
<i>UoR at SemEval-2021 Task 7: Utilizing Pre-trained DistilBERT Model and Multi-scale CNN for Humor Detection</i>	
Zehao Liu, Carl Haines and Huizhi Liang	1179
<i>TECHSSN at SemEval-2021 Task 7: Humor and Offense detection and classification using ColBERT embeddings</i>	
Rajalakshmi Sivanaiah, Angel Deborah S, S Milton Rajendram, Mirnalinee TT, Abrit Pal Singh, Aviansh Gupta and Ayush Nanda	1185
<i>Amherst685 at SemEval-2021 Task 7: Joint Modeling of Classification and Regression for Humor and Offense</i>	
Brian Zyllich, Akshay Gugnani, Gabriel Brookman and Nicholas Samoray	1190
<i>DuluthNLP at SemEval-2021 Task 7: Fine-Tuning RoBERTa Model for Humor Detection and Offense Rating</i>	
Samuel Akrah	1196
<i>CSECU-DSG at SemEval-2021 Task 7: Detecting and Rating Humor and Offense Employing Transformers</i>	
Afrin Sultana, Nabila Ayman and Abu Nowshed Chy	1204
<i>RedwoodNLP at SemEval-2021 Task 7: Ensembled Pretrained and Lightweight Models for Humor Detection</i>	
Nathan Chi and Ryan Chi	1209
<i>EndTimes at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with BERT and Ensembles</i>	
Chandan Kumar Pandey, Chirag Singh and Karan Mangla	1215
<i>IITH at SemEval-2021 Task 7: Leveraging transformer-based humorous and offensive text detection architectures using lexical and hurtlex features and task adaptive pretraining</i>	
Tathagata Raha, Ishan Sanjeev Upadhyay, Radhika Mamidi and Vasudeva Varma	1221
<i>FII FUNNY at SemEval-2021 Task 7: HaHackathon: Detecting and rating Humor and Offense</i>	
Mihai Samson and Daniela Gifu	1226
<i>Counts@IITK at SemEval-2021 Task 8: SciBERT Based Entity And Semantic Relation Extraction For Scientific Data</i>	
Akash Gangwar, Sabhay Jain, Shubham Sourav and Ashutosh Modi	1232
<i>CONNER: A Cascade Count and Measurement Extraction Tool for Scientific Discourse</i>	
Jiarun Cao, Yuejia Xiang, Yunyan Zhang, Zhiyuan Qi, Xi Chen and Yefeng Zheng	1239

<i>Stanford MLab at SemEval-2021 Task 8: 48 Hours Is All You Need</i>	
Patrick Liu, Niveditha Iyer, Erik Rozi and Ethan A. Chi .....	1245
<i>LIORI at SemEval-2021 Task 8: Ask Transformer for measurements</i>	
Adis Davletov, Denis Gordeev, Nikolay Arefyev and Emil Davletov .....	1249
<i>Sattiy at SemEval-2021 Task 9: An Ensemble Solution for Statement Verification and Evidence Finding with Tables</i>	
Xiaoyi Ruan, Meizhi Jin, Jian Ma, Haiqin Yang, Lianxin Jiang, Yang Mo and Mengyuan Zhou	1255
<i>Volta at SemEval-2021 Task 9: Statement Verification and Evidence Finding with Tables using TAPAS and Transfer Learning</i>	
Devansh Gautam, Kshitij Gupta and Manish Shrivastava .....	1262
<i>KaushikAcharya at SemEval-2021 Task 9: Candidate Generation for Fact Verification over Tables</i>	
Kaushik Acharya .....	1271
<i>AttesTable at SemEval-2021 Task 9: Extending Statement Verification with Tables for Unknown Class, and Semantic Evidence Finding</i>	
Harshit Varma, Aadish Jain, Pratik Ratadiya and Abhishek Rathi .....	1276
<i>MedAI at SemEval-2021 Task 10: Negation-aware Pre-training for Source-free Negation Detection Domain Adaptation</i>	
Jinquan Sun, Qi Zhang, Yu Wang and Lei Zhang .....	1283
<i>YNU-HPCC at SemEval-2021 Task 10: Using a Transformer-based Source-Free Domain Adaptation Model for Semantic Processing</i>	
Zhewen Yu, Jin Wang and Xuejie Zhang .....	1289
<i>ECNUICA at SemEval-2021 Task 11: Rule based Information Extraction Pipeline</i>	
Jiaju Lin, Jing Ling, Zhiwei Wang, Jiawei Liu, Qin Chen and Liang He .....	1295
<i>UOR at SemEval-2021 Task 12: On Crowd Annotations; Learning with Disagreements to optimise crowd truth</i>	
Emmanuel Osei-Brefo, Thanet Markchom and Huizhi Liang .....	1303



# Conference Program

*All times are UTC.*

## **5 August 2021**

### **14:00–15:00 Invited Talk**

*Seven Social Factors in Natural Language Processing: Theory and Practice*  
Diyi Yang

### **15:00–15:25 Plenary session: Tasks 1, 2, 4**

#### *SemEval-2021 Task 1: Lexical Complexity Prediction*

Matthew Shardlow, Richard Evans, Gustavo Henrique Paetzold and Marcos Zampieri

#### *OCHADAI-KYOTO at SemEval-2021 Task 1: Enhancing Model Generalization and Robustness for Lexical Complexity Prediction*

Yuki Taya, Lis Kanashiro Pereira, Fei Cheng and Ichiro Kobayashi

#### *SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC)*

Federico Martelli, Najla Kalach, Gabriele Tola and Roberto Navigli

#### *SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning*

Boyuan Zheng, Xiaoyu Yang, Yu-Ping Ruan, Zhenhua Ling, Quan Liu, Si Wei and Xiaodan Zhu

#### *TA-MAMC at SemEval-2021 Task 4: Task-adaptive Pretraining and Multi-head Attention for Abstract Meaning Reading Comprehension*

Jing Zhang, Yimeng Zhuang and Yinpei Su

**5 August 2021 (continued)**

**15:25–15:50 Plenary session: Tasks 5, 6, 7**

*SemEval-2021 Task 5: Toxic Spans Detection*

John Pavlopoulos, Jeffrey Sorensen, Léo Laugier and Ion Androutsopoulos

*SemEval-2021 Task 6: Detection of Persuasion Techniques in Texts and Images*

Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov and Giovanni Da San Martino

*Alpha at SemEval-2021 Task 6: Transformer Based Propaganda Classification*

Zhida Feng, Jiji Tang, Jiayang Liu, Weichong Yin, Shikun Feng, Yu Sun and Li Chen

*SemEval 2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense*

J. A. Meaney, Steven Wilson, Luis Chiruzzo, Adam Lopez and Walid Magdy

**15:50–16:00 Announcement of Best Paper Awards & General Discussion**

**16:30–17:30 Poster session: Tasks 1, 2, 4**

+ *This poster session will also be held at 02:00–03:00 and at 11:30–12:30.*

*LangResearchLab NC at SemEval-2021 Task 1: Linguistic Feature Based Modelling for Lexical Complexity*

Raksha Agarwal and Niladri Chatterjee

*Complex words identification using word-level features for SemEval-2020 Task 1*

Jenny A. Ortiz-Zambrano and Arturo Montejo-Ráez

*TUDA-CCL at SemEval-2021 Task 1: Using Gradient-boosted Regression Tree Ensembles Trained on a Heterogeneous Feature Set for Predicting Lexical Complexity*

Sebastian Gombert and Sabine Bartsch

*JCT at SemEval-2021 Task 1: Context-aware Representation for Lexical Complexity Prediction*

Chaya Liebeskind, Otniel Elkayam and Shmuel Liebeskind

**5 August 2021 (continued)**

*IAPUCP at SemEval-2021 Task 1: Stacking Fine-Tuned Transformers is Almost All You Need for Lexical Complexity Prediction*

Kervy Rivas Rojas and Fernando Alva-Manchego

*Uppsala NLP at SemEval-2021 Task 2: Multilingual Language Models for Fine-tuning and Feature Extraction in Word-in-Context Disambiguation*

Huiling You, Xingran Zhu and Sara Stymne

*SkoltechNLP at SemEval-2021 Task 2: Generating Cross-Lingual Training Data for the Word-in-Context Task*

Anton Razzhigaev, Nikolay Arefyev and Alexander Panchenko

*Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning*

Boris Zhestiankin and Maria Ponomareva

*SzegedAI at SemEval-2021 Task 2: Zero-shot Approach for Multilingual and Cross-lingual Word-in-Context Disambiguation*

Gábor Berend

*ReCAM@IITK at SemEval-2021 Task 4: BERT and ALBERT based Ensemble for Abstract Word Prediction*

Abhishek Mittal and Ashutosh Modi

*ECNU\_ICA\_1 SemEval-2021 Task 4: Leveraging Knowledge-enhanced Graph Attention Networks for Reading Comprehension of Abstract Meaning*

Pingsheng Liu, Linlin Wang, Qian Zhao, Hao Chen, Yuxi Feng, Xin Lin and liang he

*LRG at SemEval-2021 Task 4: Improving Reading Comprehension with Abstract Words using Augmentation, Linguistic Features and Voting*

Abheesht Sharma, Harshit Pandey, Gunjan Chhablani, Yash Bhartia and Tirtharaj Dash

*IIE-NLP-Eyas at SemEval-2021 Task 4: Enhancing PLM for ReCAM with Special Tokens, Re-Ranking, Siamese Encoders and Back Translation*

Yuqiang Xie, Luxi Xing, Wei Peng and Yue Hu

*NLP-IIS@UT at SemEval-2021 Task 4: Machine Reading Comprehension using the Long Document Transformer*

Hossein Basafa, Sajad Movahedi, Ali Ebrahimi, Azadeh Shakery and Hesham Faili

**5 August 2021 (continued)**

**17:30–18:30** Poster session: Tasks 5, 6, 7

+ *This poster session will also be held at 03:00–04:00 and at 12:30–13:30.*

*IITK@Detox at SemEval-2021 Task 5: Semi-Supervised Learning and Dice Loss for Toxic Spans Detection*

Archit Bansal, Abhay Kaushik and Ashutosh Modi

*UniParma at SemEval-2021 Task 5: Toxic Spans Detection Using CharacterBERT and Bag-of-Words Model*

Akbar Karimi, Leonardo Rossi and Andrea Prati

*UPB at SemEval-2021 Task 5: Virtual Adversarial Training for Toxic Spans Detection*

Andrei Paraschiv, Dumitru-Clementin Cercel and Mihai Dascalu

*NLRG at SemEval-2021 Task 5: Toxic Spans Detection Leveraging BERT-based Token Classification and Span Prediction Techniques*

Gunjan Chhablani, Abheesht Sharma, Harshit Pandey, Yash Bhartia and Shan Suthaharan

*UoB at SemEval-2021 Task 5: Extending Pre-Trained Language Models to Include Task and Domain-Specific Information for Toxic Span Prediction*

Erik Yan and Harish Tayyar Madabushi

*Cisco at SemEval-2021 Task 5: What's Toxic?: Leveraging Transformers for Multiple Toxic Span Extraction from Online Comments*

Sreyan Ghosh and Sonal Kumar

*MedAI at SemEval-2021 Task 5: Start-to-end Tagging Framework for Toxic Spans Detection*

Zhen Wang, Hongjie Fan and Junfei Liu

*HamiltonDinggg at SemEval-2021 Task 5: Investigating Toxic Span Detection using RoBERTa Pre-training*

Huiyang Ding and David Jurgens

*WVOQ at SemEval-2021 Task 6: BART for Span Detection and Classification*

Cees Roele

*HumorHunter at SemEval-2021 Task 7: Humor and Offense Recognition with Disentangled Attention*

Yubo Xie, Junze Li and Pearl Pu



## 5 August 2021 (continued)

*Grenzlinie at SemEval-2021 Task 7: Detecting and Rating Humor and Offense*

Renyuan Liu and Xiaobing Zhou

*abcbbc at SemEval-2021 Task 7: ERNIE-based Multi-task Model for Detecting and Rating Humor and Offense*

Chao Pang, Xiaoran Fan, Weiyue Su, Xuyi Chen, Shuohuan Wang, Jiaxiang Liu, Xuan Ouyang, Shikun Feng and Yu Sun

*Humor@IITK at SemEval-2021 Task 7: Large Language Models for Quantifying Humor and Offensiveness*

Aishwarya Gupta, Avik Pal, Bholeshwar Khurana, Lakshay Tyagi and Ashutosh Modi

*RoMa at SemEval-2021 Task 7: A Transformer-based Approach for Detecting and Rating Humor and Offense*

Roberto Labadie, Mariano Jason Rodriguez, Reynier Ortega and Paolo Rosso

## 6 August 2021

### 14:00–15:00 Invited Talk

*Predictability and Informativity in Communication*

Hannah Rohde

### 15:00–15:25 Plenary session: Tasks 8, 9, 12

*SemEval-2021 Task 8: MeasEval – Extracting Counts and Measurements and their Related Contexts*

Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr. and Paul Groth

*SemEval-2021 Task 9: Fact Verification and Evidence Finding for Tabular Data in Scientific Documents (SEM-TAB-FACTS)*

Nancy X. R. Wang, Diwakar Mahajan, Marina Danilevsky and Sara Rosenthal

*BreakingBERT@IITK at SemEval-2021 Task 9: Statement Verification and Evidence Finding with Tables*

Aditya Jindal, Ankur Gupta, Jaya Srivastava, Preeti Menghwani, Vijit Malik, Vishesh Kaushik and Ashutosh Modi

*SemEval-2021 Task 12: Learning with Disagreements*

Alexandra Uma, Tommaso Fornaciari, Anca Dumitrache, Tristan Miller, Jon Chamberlain, Barbara Plank, Edwin Simpson and Massimo Poesio

**6 August 2021 (continued)**

**15:25–15:50 Plenary session: Tasks 10, 11**

*SemEval-2021 Task 10: Source-Free Domain Adaptation for Semantic Processing*  
Egoitz Laparra, Xin Su, Yiyun Zhao, Özlem Uzuner, Timothy Miller and Steven Bethard

*BLCUFIGHT at SemEval-2021 Task 10: Novel Unsupervised Frameworks For Source-Free Domain Adaptation*  
Weikang Wang, Yi Wu, Yixiang Liu and Pengyuan Liu

*SemEval-2021 Task 11: NLPContributionGraph - Structuring Scholarly NLP Contributions for a Research Knowledge Graph*  
Jennifer D’Souza, Sören Auer and Ted Pedersen

*UIUC\_BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions*  
Haoyang Liu, M. Janina Sarol and Halil Kilicoglu

**15:50–16:00 Announcement of SemEval-2022 Tasks & Closing Remarks**

**16:30–17:30 Poster session: Tasks 8, 9, 12**

+ *This poster session will also be held at 02:00–03:00 and at 11:30–12:30.*

*KGP at SemEval-2021 Task 8: Leveraging Multi-Staged Language Models for Extracting Measurements, their Attributes and Relations*  
Neel Karia, Ayush Kaushal and Faraaz Mallick

*DPR at SemEval-2021 Task 8: Dynamic Path Reasoning for Measurement Relation Extraction*  
Amir Pouran Ben Veysheh, Franck Dernoncourt and Thien Huu Nguyen

*CLaC-np at SemEval-2021 Task 8: Dependency DGCNN*  
Nihatha Lathiff, Pavel PK Khloponin and Sabine Bergler

*CLaC-BP at SemEval-2021 Task 8: SciBERT Plus Rules for MeasEval*  
Benjamin Therien, Parsa Bagherzadeh and Sabine Bergler

**6 August 2021 (continued)**

*THiFly\_Queens at SemEval-2021 Task 9: Two-stage Statement Verification with Adaptive Ensembling and Slot-based Operation*

Yuxuan Zhou, Kaiyin Zhou, Xien Liu, Ji Wu and Xiaodan Zhu

*TAPAS at SemEval-2021 Task 9: Reasoning over tables with intermediate pre-training*

Thomas Müller, Julian Eisenschlos and Syrine Krichene

*BOUN at SemEval-2021 Task 9: Text Augmentation Techniques for Fact Verification in Tabular Data*

Abdullatif Köksal, Yusuf Yüksel, Bekir Yıldırım and Arzucan Özgür

**17:30–18:30 Poster session: Tasks 10, 11**

**+ *This poster session will also be held at 03:00–04:00 and at 12:30–13:30.***

*IITK at SemEval-2021 Task 10: Source-Free Unsupervised Domain Adaptation using Class Prototypes*

Harshit Kumar, Jinang Shah, Nidhi Hegde, Priyanshu Gupta, Vaibhav Jindal and Ashutosh Modi

*PTST-UoM at SemEval-2021 Task 10: Parsimonious Transfer for Sequence Tagging*

Kemal Kurniawan, Lea Frermann, Philip Schulz and Trevor Cohn

*Self-Adapter at SemEval-2021 Task 10: Entropy-based Pseudo-Labeler for Source-free Domain Adaptation*

Sangwon Yoon, Yanghoon Kim and Kyomin Jung

*The University of Arizona at SemEval-2021 Task 10: Applying Self-training, Active Learning and Data Augmentation to Source-free Domain Adaptation*

Xin Su, Yiyun Zhao and Steven Bethard

*KnowGraph@IITK at SemEval-2021 Task 11: Building Knowledge Graph for NLP Research*

Shashank Shailabh, Sajal Chaurasia and Ashutosh Modi

*YNU-HPCC at SemEval-2021 Task 11: Using a BERT Model to Extract Contributions from NLP Scholarly Articles*

Xinge Ma, Jin Wang and Xuejie Zhang

*ITNLP at SemEval-2021 Task 11: Boosting BERT with Sampling and Adversarial Training for Knowledge Extraction*

Genyu Zhang, Yu Su, Changhong He, Lei Lin, Chengjie Sun and Lili Shan

**6 August 2021 (continued)**

*Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction*

Anna Martin and Ted Pedersen

*INNOVATORS at SemEval-2021 Task-11: A Dependency Parsing and BERT-based model for Extracting Contribution Knowledge from Scientific Papers*

Hardik Arora, Tirthankar Ghosal, Sandeep Kumar, Suraj Patwal and Phil Goch

# SemEval-2021 Task 1: Lexical Complexity Prediction

Matthew Shardlow<sup>1</sup>, Richard Evans<sup>2</sup>, Gustavo Henrique Paetzold<sup>3</sup>, Marcos Zampieri<sup>4</sup>

<sup>1</sup>Manchester Metropolitan University, UK

<sup>2</sup>University of Wolverhampton, UK

<sup>3</sup>Universidade Tecnológica Federal do Paraná, Brazil

<sup>4</sup>Rochester Institute of Technology USA

m.shardlow@mmu.ac.uk

## Abstract

This paper presents the results and main findings of SemEval-2021 Task 1 - Lexical Complexity Prediction. We provided participants with an augmented version of the CompLex Corpus (Shardlow et al., 2020). CompLex is an English multi-domain corpus in which words and multi-word expressions (MWEs) were annotated with respect to their complexity using a five point Likert scale. SemEval-2021 Task 1 featured two Sub-tasks: Sub-task 1 focused on single words and Sub-task 2 focused on MWEs. The competition attracted 198 teams in total, of which 54 teams submitted official runs on the test data to Sub-task 1 and 37 to Sub-task 2.

## 1 Introduction

The occurrence of an unknown word in a sentence can adversely affect its comprehension by readers. Either they give up, misinterpret, or plough on without understanding. A committed reader may take the time to look up a word and expand their vocabulary, but even in this case they must leave the text, undermining their concentration. The natural language processing solution is to identify candidate words in a text that may be too difficult for a reader (Shardlow, 2013; Paetzold and Specia, 2016a). Each potential word is assigned a judgment by a system to determine if it was deemed ‘complex’ or not. These scores indicate which words are likely to cause problems for a reader. The words that are identified as problematic can be the subject of numerous types of intervention, such as direct replacement in the setting of lexical simplification (Gooding and Kochmar, 2019), or extra information being given in the context of explanation generation (Rello et al., 2015).

Whereas previous solutions to this task have typically considered the Complex Word Identification (CWI) task (Paetzold and Specia, 2016a; Yimam et al., 2018) in which a binary judgment of a word’s

complexity is given (i.e., is a word complex or not?), we instead focus on the Lexical Complexity Prediction (LCP) task (Shardlow et al., 2020) in which a value is assigned from a continuous scale to identify a word’s complexity (i.e., how complex is this word?). We ask multiple annotators to give a judgment on each instance in our corpus and take the average prediction as our complexity label. The former task (CWI) forces each user to make a subjective judgment about the nature of the word that models their personal vocabulary. Many factors may affect the annotator’s judgment including their education level, first language, specialism or familiarity with the text at hand. The annotators may also disagree on the level of difficulty at which to label a word as complex. One annotator may label every word they feel is above average difficulty, another may label words that they feel unfamiliar with, but understand from the context, whereas another annotator may only label those words that they find totally incomprehensible, even in context. Our introduction of the LCP task seeks to address this annotator confusion by giving annotators a Likert scale to provide their judgments. Whilst annotators must still give a subjective judgment depending on their own understanding, familiarity and vocabulary — they do so in a way that better captures the meaning behind each judgment they have given. By aggregating these judgments we have developed a dataset that contains continuous labels in the range of 0–1 for each instance. This means that rather than a system predicting whether a word is complex or not (0 or 1), instead a system must now predict where, on our continuous scale, a word falls (0–1).

Consider the following sentence taken from a biomedical source, where the target word ‘observation’ has been highlighted:

- (1) The **observation** of unequal expression leads to a number of questions.

In the binary annotation setting of CWI some annotators may rightly consider this term non-complex, whereas others may rightly consider it to be complex. Whilst the meaning of the word is reasonably clear to someone with scientific training, the context in which it is used is unfamiliar for a lay reader and will likely lead to them considering it complex. In our new LCP setting, we are able to ask annotators to mark the word on a scale from very easy to very difficult. Each user can give their subjective interpretation on this scale indicating how difficult they found the word. Whilst annotators will inevitably disagree (some finding it more or less difficult), this is captured and quantified as part of our annotations, with a word of this type likely to lead to a medium complexity value.

LCP is useful as part of the wider task of lexical simplification (Devlin and Tait, 1998), where it can be used to both identify candidate words for simplification (Shardlow, 2013) and rank potential words as replacements (Paetzold and Specia, 2017). LCP is also relevant to the field of readability assessment, where knowing the proportion of complex words in a text helps to identify the overall complexity of the text (Dale and Chall., 1948).

This paper presents SemEval-2021 Task 1: Lexical Complexity Prediction. In this task we developed a new dataset for complexity prediction based on the previously published CompLex dataset. Our dataset covers 10,800 instances spanning 3 genres and containing unigrams and bigrams as targets for complexity prediction. We solicited participants in our task and released a trial, training and test split in accordance with the SemEval schedule. We accepted submissions in two separate Sub-tasks, the first being single words only and the second taking single words and multi-word expressions (modelled by our bigrams). In total 55 teams participated across the two Sub-tasks.

The rest of this paper is structured as follows: In Section 2 we discuss the previous two iterations of the CWI task. In Section 3, we present the CompLex 2.0 dataset that we have used for our task, including the methodology we used to produce trial, test and training splits. In Section 5, we show the results of the participating systems and compare the features that were used by each system. We finally discuss the nature of LCP in Section 7 and give concluding remarks in Section 8

## 2 Related Tasks

**CWI 2016 at SemEval** The CWI shared task was organized at SemEval 2016 (Paetzold and Specia, 2016a). The CWI 2016 organizers introduced a new CWI dataset and reported the results of 42 CWI systems developed by 21 teams. Words in their dataset were considered complex if they were difficult to understand for non-native English speakers according to a binary labelling protocol. A word was considered complex if at least one of the annotators found it to be difficult. The training dataset consisted of 2,237 instances, each labelled by 20 annotators and the test dataset had 88,221 instances, each labelled by 1 annotator (Paetzold and Specia, 2016a).

The participating systems leveraged lexical features (Choubey and Pateria, 2016; Bingel et al., 2016; Quijada and Medero, 2016) and word embeddings (Kuru, 2016; S.P et al., 2016; Gillin, 2016), as well as finding that frequency features, such as those taken from Wikipedia (Konkol, 2016; Wróbel, 2016) were useful. Systems used binary classifiers such as SVMs (Kuru, 2016; S.P et al., 2016; Choubey and Pateria, 2016), Decision Trees (Choubey and Pateria, 2016; Quijada and Medero, 2016; Malmasi et al., 2016), Random Forests (Ronzano et al., 2016; Brooke et al., 2016; Zampieri et al., 2016; Mukherjee et al., 2016) and threshold-based metrics (Kauchak, 2016; Wróbel, 2016) to predict the complexity labels. The winning system made use of threshold-based methods and features extracted from Simple Wikipedia (Paetzold and Specia, 2016b).

A post-competition analysis (Zampieri et al., 2017) with oracle and ensemble methods showed that most systems performed poorly due mostly to the way in which the data was annotated and the the small size of the training dataset.

**CWI 2018 at BEA** The second CWI Shared Task was organized at the BEA workshop 2018 (Yimam et al., 2018). Unlike the first task, this second task had two objectives. The first objective was the binary complex or non-complex classification of target words. The second objective was regression or probabilistic classification in which 13 teams were asked to assign the probability of a target word being considered complex by a set of language learners. A major difference in this second task was that datasets of differing genres: (TEXT GENRES) as well as English, German and Spanish datasets

for monolingual speakers and a French dataset for multilingual speakers were provided (Yimam et al., 2018).

Similar to 2016, systems made use of a variety of lexical features including word length (Wani et al., 2018; De Hertog and Tack, 2018; AbuRa’ed and Saggion, 2018; Hartmann and dos Santos, 2018; Alfter and Pilán, 2018; Kajiwara and Komachi, 2018), frequency (De Hertog and Tack, 2018; Aroyehun et al., 2018; Alfter and Pilán, 2018; Kajiwara and Komachi, 2018), N-gram features (Gooding and Kochmar, 2018; Popović, 2018; Hartmann and dos Santos, 2018; Alfter and Pilán, 2018; Butnaru and Ionescu, 2018) and word embeddings (De Hertog and Tack, 2018; AbuRa’ed and Saggion, 2018; Aroyehun et al., 2018; Butnaru and Ionescu, 2018). A variety of classifiers were used ranging from traditional machine learning classifiers (Gooding and Kochmar, 2018; Popović, 2018; AbuRa’ed and Saggion, 2018), to Neural Networks (De Hertog and Tack, 2018; Aroyehun et al., 2018). The winning system made use of Adaboost with WordNet features, POS tags, dependency parsing relations and psycholinguistic features (Gooding and Kochmar, 2018).

### 3 Data

We previously reported on the annotation of the CompLex dataset (Shardlow et al., 2020) (hereafter referred to as CompLex 1.0), in which we annotated around 10,000 instances for lexical complexity using the Figure Eight platform. The instances spanned three genres: **Europarl**, taken from the proceedings of the European Parliament (Koehn, 2005); **The Bible**, taken from an electronic distribution of the World English Bible translation (Christodouloupoulos and Steedman, 2015) and **Biomedical** literature, taken from the CRAFT corpus (Bada et al., 2012). We limited our annotations to focus only on nouns and multi-word expressions following a Noun-Noun or Adjective-Noun pattern, using the POS tagger from Stanford CoreNLP (Manning et al., 2014) to identify these patterns.

Whilst these annotations allowed us to report on the dataset and to show some trends, the overall quality of the annotations we received was poor and we ended up discarding a large number of the annotations. For CompLex 1.0 we retained only instances with four or more annotations and the low number of annotations (average number of annotators = 7) led to the overall dataset being less

reliable than initially expected

For the Shared Task we chose to boost the number of annotations on the same data as used for CompLex 1.0 using Amazon’s Mechanical Turk platform. We requested a further 10 annotations on each data instance bringing up the average number of annotators per instance. Annotators were presented with the same task layout as in the annotation of CompLex 1.0 and we defined the Likert Scale points as previously:

**Very Easy:** Words which were very familiar to an annotator.

**Easy:** Words with which an annotator was aware of the meaning.

**Neutral:** A word which was neither difficult nor easy.

**Difficult:** Words which an annotator was unclear of the meaning, but may have been able to infer the meaning from the sentence.

**Very Difficult:** Words that an annotator had never seen before, or were very unclear.

These annotations were aggregated with the retained annotations of CompLex 1.0 to give our new dataset, CompLex 2.0, covering 10,800 instances across single and multi-words and across 3 genres.

The features that make our corpus distinct from other corpora which focus on the CWI and LCP tasks are described below:

**Continuous Annotations:** We have annotated our data using a 5-point Likert Scale. Each instance has been annotated multiple times and we have taken the mean average of these annotations as the label for each data instance. To calculate this average we converted the Likert Scale points to a continuous scale as follows: Very Easy  $\rightarrow$  0, Easy  $\rightarrow$  0.25, Neutral  $\rightarrow$  0.5, Difficult  $\rightarrow$  0.75, Very Difficult  $\rightarrow$  1.0.

**Contextual Annotations:** Each instance in the corpus is presented with its enclosing sentence as context. This ensures that the sense of a word can be identified when assigning it a complexity value. Whereas previous work has reannotated the data from the CWI–2018 shared task with word senses (Strohmaier et al., 2020), we do not make explicit sense distinctions between our tokens, instead leaving this task up to participants.

**Repeated Token Instances:** We provide more than one context for each token (up to a maximum of five contexts per genre). These words were annotated separately during annotation, with the expectation that tokens in different contexts would receive differing complexity values. This deliberately penalises systems that do not take the context of a word into account.

**Multi-word Expressions:** In our corpus we have provided 1,800 instances of multi-word expressions (split across our 3 sub-corpora). Each MWE is modelled as a Noun-Noun or Adjective-Noun pattern followed by any POS tag which is not a noun. This avoids selecting the first portion of complex noun phrases. There is no guarantee that these will correspond to true MWEs that take on a meaning beyond the sum of their parts, and further investigation into the types of MWEs present in the corpus would be informative.

**Aggregated Annotations:** By aggregating the Likert scale labels we have generated crowd-sourced complexity labels for each instance in our corpus. We are assuming that, although there is inevitably some noise in any large annotation project (and especially so in crowd-sourcing), this will even out in the averaging process to give a mean value reflecting the appropriate complexity for each instance. By taking the mean average we are assuming unimodal distributions in our annotations.

**Varied Genres:** We have selected for diverse genres as mentioned above. Previous CWI datasets have focused on informal text such as Wikipedia and multi-genre text, such as news. By focusing on specific texts we force systems to learn generalised complexity annotations that are appropriate in a cross-genre setting.

We have presented summary statistics for CompLex 2.0 in Table 1. In total, 5,617 unique words are split across 10,800 contexts, with an average complexity across our entire dataset of 0.321. Each genre has 3,600 contexts, with each split between 3,000 single words and 600 multi-word expressions. Whereas single words are slightly below the average complexity of the dataset at 0.302, multi-word expressions are much more complex at 0.419,

indicating that annotators found these more difficult to understand. Similarly Europarl and the Bible were less complex than the corpus average, whereas the Biomedical articles were more complex. The number of unique tokens varies from one genre to another as the tokens were selected at random and discarded if there were already more than 5 occurrences of the given token already in the dataset. This stochastic selection process led to a varied dataset with some tokens only having one context, whereas others have as many as five in a given genre. On average each token has around 2 contexts.

## 4 Data Splits

In order to run the shared task we partitioned our dataset into Trial, Train and Test splits and distributed these according to the SemEval schedule. A criticism of previous CWI shared tasks is that the training data did not accurately reflect the distribution of instances in the testing data. We sought to avoid this by stratifying our selection process for a number of factors. The first factor we considered was genre. We ensured that an even number of instances from each genre was present in each split. We also stratified for complexity, ensuring that each split had a similar distribution of complexities. Finally we also stratified the splits by token, ensuring that multiple instances containing the same token occurred in only one split. This last criterion ensures that systems do not overfit to the test data by learning the complexities of specific tokens in the training data.

Performing a robust stratification of a dataset according to multiple features is a non-trivial optimisation problem. We solved this by first grouping all instances in a genre by token and sorting these groups by the complexity of the least complex instance in the group. For each genre, we passed through this sorted list and for each set of 20 groups we put the first group in the trial set, the next two groups in the test set and the remaining 17 groups in the training data. This allowed us to get a rough 5-85-10 split between trial, training and test data. The trial and training data were released in this ordered format, however to prevent systems from guessing the labels based on the data ordering we randomised the order of the instances in the test data prior to release. The splits that we used for the Shared Task are available via GitHub<sup>1</sup>.

<sup>1</sup><https://github.com/MMU-TDMLab/CompLex>



Subset	Genre	Contexts	Unique Tokens	Average Complexity
All	<b>Total</b>	<b>10,800</b>	<b>5,617</b>	<b>0.321</b>
	Europarl	3,600	2,227	0.303
	Biomed	3,600	1,904	0.353
	Bible	3,600	1,934	0.307
Single	<b>Total</b>	<b>9,000</b>	<b>4,129</b>	<b>0.302</b>
	Europarl	3,000	1,725	0.286
	Biomed	3,000	1,388	0.325
	Bible	3,000	1,462	0.293
MWE	<b>Total</b>	<b>1,800</b>	<b>1,488</b>	<b>0.419</b>
	Europarl	600	502	0.388
	Biomed	600	516	0.491
	Bible	600	472	0.377

Table 1: The statistics for CompLex 2.0.

Table 2 presents statistics on each split in our data, where it can be seen that we were able to achieve a roughly even split between genres across the trial, train and test data.

Subset	Genre	Trial	Train	Test
All	<b>Total</b>	<b>520</b>	<b>9179</b>	<b>1101</b>
	Europarl	180	3010	410
	Biomed	168	3090	342
	Bible	172	3079	349
Single	<b>Total</b>	421	7662	917
	Europarl	143	2512	345
	Biomed	135	2576	289
	Bible	143	2574	283
MWE	<b>Total</b>	99	1517	184
	Europarl	37	498	65
	Biomed	33	514	53
	Bible	29	505	66

Table 2: The Trial, Train and Test splits that were used as part of the shared task.

## 5 Results

The full results of our task can be seen in Appendix A. We had 55 teams participate in our 2 Sub-tasks, with 19 participating in Sub-task 1 only, 1 participating in Sub-task 2 only and 36 participating in both Sub-tasks. We have used Pearson’s correlation for our final ranking of participants, but we have also included other metrics that are appropriate for evaluating continuous and ranked data and provided secondary rankings of these.

Sub-task 1 asked participants to assign complexity values to each of the single words instances in our corpus. For Sub-task 2, we asked participants to submit results on both single words and MWEs. We did not rank participants on MWE-only submis-

sions due to the relatively small number of MWEs in our corpus (184 in the test set).

The metrics we chose for ranking were as follows:

**Pearson’s Correlation:** We chose this metric as our primary method of ranking as it is well known and understood, especially in the context of evaluating systems with continuous outputs. Pearson’s correlation is robust to changes in scale and measures how the input variables change with each other.

**Spearman’s Rank:** This metric does not consider the values output by a system, or in the test labels, only the order of those labels. It was chosen as a secondary metric as it is more robust to outliers than Pearson’s correlation.

**Mean Absolute Error (MAE):** Typically used for the evaluation of regression tasks, we included MAE as it gives an indication of how close the predicted labels were to the gold labels for our task.

**Mean Squared Error (MSE):** There is little difference in the calculation of MSE vs. MAE, however we also include this metric for completeness.

**R2:** This measures the proportion of variance of the original labels captured by the predicted labels. It is possible to do well on all the other metrics, yet do poorly on R2 if a system produces annotations with a different distribution than those in the original labels.

In Table 3 we show the scores of the top 10 systems across our 2 Sub-tasks according to Pearson’s

Team	Task 1	
	Pearson	R2
JUST BLUE	0.7886 (1)	0.6172 (2)
DeepBlueAI	0.7882 (2)	0.6210 (1)
Alejandro Mosquera	0.7790 (3)	0.6062 (3)
Andi	0.7782 (4)	0.6036 (4)
CS-UM6P	0.7779 (5)	0.3813 (47)
tuqa	0.7772 (6)	0.5771 (12)
OCHADAI-KYOTO	0.7772 (7)	0.6015 (5)
BigGreen	0.7749 (8)	0.5983 (6)
CSECU-DSC	0.7716 (9)	0.5909 (8)
IA PUCP	0.7704 (10)	0.5929 (7)
<i>Frequency Baseline</i>	0.5287	0.2779
Task 2		
DeepBlueAI	0.8612 (1)	0.7389 (1)
rg_pa	0.8575 (2)	0.7035 (5)
xiang_wen_tian	0.8571 (3)	0.7012 (7)
andi_gpu	0.8543 (4)	0.7055 (4)
ren_wo_xing	0.8541 (5)	0.6967 (8)
Andi	0.8506 (6)	0.7107 (2)
CS-UM6P	0.8489 (7)	0.6380 (17)
OCHADAI-KYOTO	0.8438 (8)	0.7103 (3)
LAST	0.8417 (9)	0.7030 (6)
KFU	0.8406 (10)	0.6967 (9)
<i>Frequency Baseline</i>	0.6571	0.4030

Table 3: The top 10 systems for each task according to Pearson’s correlation. We have also included R2 score to help interpret the former. For full rankings, see Appendix A

Correlation. We have only reported on Pearson’s correlation and R2 in these tables, but the full results with all metrics are available in Appendix A. We have included a Frequency Baseline produced using log-frequency from the Google Web1T and linear regression, which was beaten by the majority of our systems. From these results we can see that systems were able to attain reasonably high scores on our dataset, with the winning systems reporting Pearson’s Correlation of 0.7886 for Sub-task 1 and 0.8612 for Sub-task 2, as well as high R2 scores of 0.6210 for Sub-task 1 and 0.7389 for Sub-task 2. The rankings remained stable across Spearman’s rank, MAE and MSE, with some small variations. Scores were generally higher on Sub-task 2 than on Sub-task 1, and this is likely to be because of the different groups of token-types (single words and MWEs). MWEs are known to be more complex than single words and so this fact may have implicitly helped systems to better model the variance of complexities between the two groups.

## 6 Participating Systems

In this section we have analysed the participating systems in our task. System Description papers were submitted by 32 teams. In the subsections below, we have first given brief summaries of some of the top systems according to Pearson’s correlation for each task for which we had a description. We then discuss the features used across different systems, as well as the approaches to the task that different teams chose to take. We have prepared a comprehensive table comparing the features and approaches of all systems for which we have the relevant information in Appendix B.

### 6.1 System Summaries

**DeepBlueAI:** This system attained the highest Pearson’s Correlation on Sub-task 2 and the second highest Pearson’s Correlation on Sub-task 1. It also attained the highest R2 score across both tasks. The system used an ensemble of pre-trained language models fine-tuned for the task with Pseudo Labelling, Data Augmentation, Stacked Training Models and Multi-Sample Dropout. The data was encoded for the transformer models using the genre and token as a query string and the given context as a supplementary input.

**JUST BLUE:** This system attained the highest Pearson’s Correlation for Sub-task 1. The system did not participate in Sub-task 2. This system makes use of an ensemble of BERT and RoBERTa. Separate models are fine-tuned for context and token prediction and these are weighted 20-80 respectively. The average of the BERT models and RoBERTa models is taken to give a final score.

**RG PA:** This system attained the second highest Pearson’s Correlation for Sub-task 2. The system uses a fine-tuned RoBERTa model and boosts the training data for the second task by identifying similar examples from the single-word portion of the dataset to train the multi-word classifier. They use an ensemble of RoBERTa models in their final classification, averaging the outputs to enhance performance.

**Alejandro Mosquera:** This system attained the third highest Pearson’s Correlation for Sub-task 1. The system used a feature-based approach, incorporating length, frequency, semantic features from WordNet and sentence level readability features. These were passed through a Gradient Boosted Regression.

**Andi:** This system attained the fourth highest Pearson’s Correlation for Sub-task 1. They combine a traditional feature based approach with features from pre-trained language models. They use psycholinguistic features, as well as GLoVE and Word2Vec Embeddings. They also take features from an ensemble of Language models: BERT, RoBERTa, ELECTRA, ALBERT, DeBERTa. All features are passed through Gradient Boosted Regression to give the final output score.

**CS-UM6P:** This system attained the fifth highest Pearson’s Correlation for Sub-task 1 and the seventh highest Pearson’s Correlation for Sub-task 2. The system uses BERT and RoBERTa and encodes the context and token for the language models to learn from. Interestingly, whilst this system scored highly for Pearson’s correlation the R2 metric is much lower on both Sub-tasks. This may indicate the presence of significant outliers in the system’s output.

**OCHADAI-KYOTO:** This system attained the seventh highest Pearson’s Correlation on Sub-task 1 and the eight highest Pearson’s Correlation on Sub-task 2. The system used a fine-tuned BERT and RoBERTa model with the token and context encoded. They employed multiple training strategies to boost performance.

## 6.2 Approaches

There are three main types of systems that were submitted to our task. In line with the state of the art in modern NLP, these can be categorised as: Feature-based systems, Deep Learning Systems and Systems which use a hybrid of the former two approaches. Although Deep Learning Based systems have attained the highest Pearson’s Correlation on both Sub-tasks, occupying the first two places in each task, Feature based systems are not far behind, attaining the third and fourth spots on Sub-task 1 with a similar score to the top systems. We have described each approach as applied to our task below.

Feature-based systems use a variety of features known to be useful for lexical complexity. In particular, lexical frequency and word length feature heavily with many different ways of calculating these metrics such as looking at various corpora and investigating syllable or morpheme length. Psycholinguistic features which model people’s perception of words are understandably popular for this task as complexity is a perceived phenomenon.

Semantic features taken from WordNet modelling the sense of the word and it’s ambiguity or abstractness have been used widely, as well as sentence level features aiming to model the context around the target words. Some systems chose to identify named entities, as these may be innately more difficult for a reader. Word inclusion lists were also a popular feature, denoting whether a word was found on a given list of easy to read vocabulary. Finally, word embeddings are a popular feature, coming from static resources such as GLoVE or Word2Vec, but also being derived through the use of Transformer models such as BERT, RoBERTa, XLNet or GPT-2, which provide context dependent embeddings suitable for our task.

These features are passed through a regression system, with Gradient Boosted Regression and Random Forest Regression being two popular approaches amongst participants for this task. Both apply scale invariance meaning that less pre-processing of inputs is necessary.

Deep Learning Based systems invariably rely on a pre-trained language model and fine-tune this using transfer learning to attain strong scores on the task. BERT and RoBERTa were used widely in our task, with some participants also opting for ALBERT, ERNIE, or other such language models. To prepare data for these language models, most participants following this approach concatenated the token with the context, separated by a special token ( $\langle SEP \rangle$ ). The Language Model was then trained and the embedding of the  $\langle CLS \rangle$  token extracted and passed through a further fine-tuned network for complexity prediction. Adaptations to this methodology include applying training strategies such as adversarial training, multi-task learning, dummy annotation generation and capsule networks.

Finally, hybrid approaches use a mixture of Deep Learning by fine-tuning a neural network alongside feature-based approaches. The features may be concatenated to the input embeddings, or may be concatenated at the output prior to further training. Whilst this strategy appears to be the best of both worlds, uniting linguistic knowledge with the power of pre-trained language models, the hybrid systems do not tend to perform as well as either feature based or deep learning systems.

## 6.3 MWEs

For Sub-task 2 we asked participants to submit both predictions for single words and multi-words

from our corpus. We hoped this would encourage participants to consider models that adapted single word lexical complexity to multi-word lexical complexity. We observed a number of strategies that participants employed to create the annotations for this secondary portion of our data.

For systems that employed a deep learning approach, it was relatively simple to incorporate MWEs as part of their training procedure. These systems encoded the input using a query and context, separated by a  $\langle SEP \rangle$  token. The number of tokens prior to the  $\langle SEP \rangle$  token did not matter and either one or two tokens could be placed there to handle single and multi-word instances simultaneously.

However, feature based systems could not employ this trick and needed to devise more imaginative strategies for handling MWEs. Some systems handled them by averaging the features of both tokens in the MWE, or by predicting scores for each token and then averaging these scores. Other systems doubled their feature space for MWEs and trained a new model which took the features of both words into account.

## 7 Discussion

In this paper we have posited the new task of Lexical Complexity Prediction. This builds on previous work on Complex Word Identification, specifically by providing annotations which are continuous rather than binary or probabilistic as in previous tasks. Additionally, we provided a dataset with annotations in context, covering three diverse genres and incorporating MWEs, as well as single tokens. We have moved towards this task, rather than rerunning another CWI task as the outputs of the models are more useful for a diverse range of follow-on tasks. For example, whereas CWI is particularly useful as a preprocessing step for Lexical simplification (identifying which words should be transformed), LCP may also be useful for readability assessment or as a rich feature in other downstream NLP tasks. A continuous annotation allows a ranking to be given over words, rather than binary categories, meaning that we can not only tell whether a word is likely to be difficult for a reader, but also how difficult that word is likely to be. If a system requires binary complexity (as in the case of lexical simplification) it is easy to transform our continuous complexity values into a binary value by placing a threshold on the complex-

ity scale. The value of the threshold to be selected will likely depend on the target audience, with more competent speakers requiring a higher threshold. When selecting a threshold, the categories we used for annotation should be taken into account, so for example a threshold of 0.5 would indicate all words that were rated as neutral or above.

To create our annotated dataset, we employed crowdsourcing with a Likert scale and aggregated the categorical judgments on this scale to give a continuous annotation. It should be noted that this is not the same as giving a truly continuous judgment (i.e., asking each annotator to give a value between 0 and 1). We selected this protocol as the Likert Scale is familiar to annotators and allows them to select according to defined points (we provided the definitions given earlier at annotation time). The annotation points that we gave were not intended to give an even distribution of annotations and it was our expectation that most words would be familiar to some degree, falling in the very easy or easy categories. We pre-selected for harder words to ensure that there were also words in the difficult and very difficult categories. As such, the corpus we have presented is not designed to be representative of the distribution of words across the English language. To create such a corpus, one would need to annotate all words according to our scale with no filtering. The general distribution of annotations in our corpus is towards the easier end of the Likert scale.

A criticism of the approach we have employed is that it allows for subjectivity in the annotation process. Certainly one annotator’s perception of complexity will be different to another’s. Giving fixed values of complexity for every word will not reflect the specific difficulties that one reader, or one reader group will face. The annotations we have provided are averaged values of the annotations given by our annotators, we chose to keep all instances, rather than filtering out those where annotators gave a wide spread of complexity annotations. Further work may be undertaken to give interesting insights into the nature of subjectivity in annotations. For example, some words may be rated as easy or difficult by all annotators, whereas others may receive both easy and difficult annotations, indicating that the perceived complexity of the instance is more subjective. We did not make the individual annotations available as part of the shared task data, to encourage systems to focus

primarily on the prediction of complexity.

An issue with the previous shared tasks is that scores were typically low and that systems tended to struggle to beat reasonable baselines, such as those based on lexical frequency. We were pleased to see that systems participating in our task returned scores that indicated that they had learnt to model the problem well (Pearson’s Correlation of 0.7886 on Task 1 and 0.8612 on Task 2). MWEs are typically more complex than single words and it may be the case that these exhibited a lower variance, and were thus easier to predict for the systems. The strong Pearson’s Correlation is backed up by a high R2 score (0.6172 for Task 1 and 0.7389 for Task 2), which indicates that the variance in the data is captured accurately by the models’ predictions. These models strongly outperformed a reasonable baseline based on word frequency as shown in Table 3.

Whilst we have chosen in this report to rank systems based on their score on Pearson’s correlation, giving a final ranking over all systems, it should be noted that there is very little variation in score between the top systems and all other systems. For Task 1 there are 0.0182 points of Pearson’s Correlation separating the systems at ranks 1 and 10. For Task 2 a similar difference of 0.021 points of Pearson’s Correlation separates the systems at ranks 1 and 10. These are small differences and it may be the case that had we selected a different random split in our dataset this would have led to a different ordering in our results (Gorman and Bedrick, 2019; Søgaard et al., 2020). This is not unique to our task and is something for the SemEval community to ruminate on as the focus of NLP tasks continues to move towards better evaluation rather than better systems.

An analysis of the systems that participated in our task showed that there was little variation between Deep Learning approaches and Feature Based approaches, although Deep Learning approaches ultimately attained the highest scores on our data. Generally the Deep Learning and Feature Based approaches are interleaved in our results table, showing that both approaches are still relevant for LCP. One factor that did appear to affect system output was the inclusion of context, whether that was in a deep learning setting or a feature based setting. Systems which reported using no context appeared to perform worse in the overall rankings. Another feature that may have helped performance

is the inclusion of previous CWI datasets (Yimam et al., 2017; Maddela and Xu, 2018). We were aware of these when developing the corpus and attempted to make our data sufficiently distinct in style to prevent direct reuse of these resources.

A limitation of our task is that it focuses solely on LCP for the English Language. Previous CWI shared tasks (Yimam et al., 2018) and simplification efforts (Saggion et al., 2015; Aluísio and Gasperin, 2010) have focused on languages other than English and we hope to extend this task in the future to other languages.

## 8 Conclusion

We have presented the SemEval-2021 Task 1 on Lexical Complexity Prediction. We developed a new dataset focusing on continuous annotations in context across three genres. We solicited participants via SemEval and 55 teams submitted results across our two Sub-tasks. We have shown the results of these systems and discussed the factors that helped systems to perform well. We have analysed all the systems that participated and categorised their findings to help future researchers understand which approaches are suitable for LCP.

## References

- Ahmed AbuRa’ed and Horacio Saggion. 2018. LaS-TUS/TALN at Complex Word Identification (CWI) 2018 Shared Task. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- David Alfter and Ildikó Pilán. 2018. SB@GU at the Complex Word Identification 2018 Shared Task. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Sandra Aluísio and Caroline Gasperin. 2010. *Fostering digital inclusion and accessibility: The PorSimples project for simplification of Portuguese texts*. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53, Los Angeles, California. Association for Computational Linguistics.
- Segun Taofeek Aroyehun, Jason Angel, Daniel Alejandro Pérez Alvarez, and Alexander Gelbukh. 2018. Complex Word Identification: Convolutional Neural Network vs. Feature Engineering . In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans,

- United States. Association for Computational Linguistics.
- Abdul Aziz, MD, Akram Hossain, and Abu Nowshed Chy. 2021. CSECU-DSG at SemEval-2021 Task 1: Fusion of Transformer Models for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Brettonel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):1–20.
- Yves Bestgen. 2021. LAST at SemEval-2021 Task 1: Improving Multi-Word Complexity Prediction Using Bigram Association Measures. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Joachim Bingel, Natalie Schluter, and Héctor Martínez Alonso. 2016. CoastalCPH at SemEval-2016 Task 11: The importance of designing your Neural Networks right. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1028–1033, San Diego, California. Association for Computational Linguistics.
- Julian Brooke, Alexandra Uitdenbogerd, and Timothy Baldwin. 2016. Melbourne at SemEval 2016 Task 11: Classifying Type-level Word Complexity using Random Forests with Corpus and Word List Features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 975–981, San Diego, California. Association for Computational Linguistics.
- Andrei Butnaru and Radu Tudor Ionescu. 2018. UnibucKernel: A kernel-based learning method for complex word identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Prafulla Choubey and Shubham Pateria. 2016. Garuda & Bhasha at SemEval-2016 Task 11: Complex Word Identification Using Aggregated Learning Models. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1006–1010, San Diego, California. Association for Computational Linguistics.
- Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.
- E. Dale and J. S. Chall. 1948. A formula for predicting readability. *Educational research bulletin*, 27.
- Dirk De Hertog and Anaïs Tack. 2018. Deep Learning Architecture for Complex Word Identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Abhinandan Desai, Kai North, Marcos Zampieri, and Christopher Homan. 2021. LCP-RIT at SemEval-2021 Task 1: Exploring Linguistic Features for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- S. Devlin and J. Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. In *Linguistic Databases*. Stanford, USA: CSLI Publications.
- Robert Flynn and Matthew Shardlow. 2021. Manchester Metropolitan at SemEval-2021 Task 1: Convolutional Networks for Complex Word Identification. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Nat Gillin. 2016. Sensible at SemEval-2016 Task 11: Neural Nonsense Mangled in Ensemble Mess. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 963–968, San Diego, California. Association for Computational Linguistics.
- Sebastian Gombert and Sabine Bartsch. 2021. TUDA-CCL at SemEval-2021 Task 1: Using Gradient-boosted Regression Tree Ensembles Trained on a Heterogeneous Feature Set for Predicting Lexical Complexity. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2019. Recursive context-aware lexical simplification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4855–4865.
- Kyle Gorman and Steven Bedrick. 2019. We need to talk about standard splits. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2786–2791, Florence, Italy. Association for Computational Linguistics.
- Nathan Hartmann and Leandro Borges dos Santos. 2018. NILC at CWI 2018: Exploring Feature Engineering and Feature Learning. In *Proceedings of the*

- 13th Workshop on Innovative Use of NLP for Building Educational Applications, New Orleans, United States. Association for Computational Linguistics.
- Bo Huang, Yang Bai, and Xiaobing Zhou. 2021. hub at SemEval-2021 Task 1: Fusion of Sentence and Word Frequency to Predict Lexical Complexity. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Aadil Islam, Weicheng Ma, and Soroush Vosoughi. 2021. BigGreen at SemEval-2021 Task 1: Lexical Complexity Prediction with Assembly Models. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Tomoyuki Kajiwaru and Mamoru Komachi. 2018. Complex Word Identification Based on Frequency in a Learner Corpus. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- David Kauchak. 2016. Pomona at SemEval-2016 Task 11: Predicting Word Complexity Based on Corpus Frequency. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1047–1051, San Diego, California. Association for Computational Linguistics.
- Milton King, Ali Hakimi Parizi, Samin Fakharian, and Paul Cook. 2021. UNBNLP at SemEval-2021 Task 1: Predicting lexical complexity with masked language models and character-level encoders. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.
- Michal Konkol. 2016. UWB at SemEval-2016 Task 11: Exploring Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1038–1041, San Diego, California. Association for Computational Linguistics.
- Onur Kuru. 2016. AI-KU at SemEval-2016 Task 11: Word Embeddings and Substring Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1042–1046, San Diego, California. Association for Computational Linguistics.
- Chaya Liebeskind, Otniel Elkayam, and Shmuel Liebeskind. 2021. JCT at SemEval-2021 Task 1: Context-aware Representation for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Mounica Maddela and Wei Xu. 2018. A word-complexity lexicon and a neural readability ranking model for lexical simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3749–3760.
- Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. LTG at SemEval-2016 Task 11: Complex Word Identification with Classifier Ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000, San Diego, California. Association for Computational Linguistics.
- Nabil El Mamoun, Abdelkader El Mahdaouy, Abdelah El Mekki, Kabil Essefar, and Ismail Berrada. 2021. CS-UM6P at SemEval-2021 Task 1: A Deep Learning Model-based Pre-trained Transformer Encoder for Lexical Complexity. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.
- Alejandro Mosquera. 2021. Alejandro Mosquera at SemEval-2021 Task 1: Exploring Sentence and Word Features for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. JU\_NLP at SemEval-2016 Task 11: Identifying Complex Words in a Sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 986–990, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016a. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016b. SV000gg at SemEval-2016 Task 11: Heavy Gauge Complex Word Identification with System Voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the*

- Association for Computational Linguistics: Volume 2, Short Papers*, pages 34–40, Valencia, Spain. Association for Computational Linguistics.
- Gustavo Henrique Paetzold. 2021. UTFPR at SemEval-2021 Task 1: Complexity Prediction by Combining BERT Vectors and Classic Features. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. 2021. DeepBlueAI at SemEval-2021 Task 1: Lexical Complexity Prediction with A Deep Ensemble Approach. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Maja Popović. 2018. Complex Word Identification using Character n-grams. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Maury Quijada and Julie Medero. 2016. [HMC at SemEval-2016 Task 11: Identifying Complex Words Using Depth-limited Decision Trees](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1034–1037, San Diego, California. Association for Computational Linguistics.
- Gang Rao, Maochang Li, Xiaolong Hou, Lianxin Jiang, Yang Mo, and Jianping Shen. 2021. RG PA at SemEval-2021 Task 1: A Contextual Attention-based Model with RoBERTa for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Luz Rello, Roberto Carlini, Ricardo Baeza-Yates, and Jeffrey P Bigham. 2015. A plug-in to aid online reading in spanish. In *Proceedings of the 12th Web for All Conference*, pages 1–4.
- Kervy Rivas Rojas and Fernando Alva-Manchego. 2021. IAPUCP at SemEval-2021 Task 1: Stacking Fine-Tuned Transformers is Almost All You Need for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Francesco Ronzano, Ahmed Abura’ed, Luis Espinosa Anke, and Horacio Saggion. 2016. [TALN at SemEval-2016 Task 11: Modelling Complex Words by Contextual, Lexical and Semantic Features](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1011–1016, San Diego, California. Association for Computational Linguistics.
- Armand Rotaru. 2021. ANDI at SemEval-2021 Task 1: Predicting complexity in context using distributional models, behavioural norms, and lexical resources. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Erik Rozi, Niveditha Iyer, Gordon Chi, Enok Choe, Kathy Lee, Kevin Liu, Patrick Liu, Zander Lack, Jillian Tang, and Ethan A. Chi. 2021. Stanford MLab at SemEval-2021 Task 1: Tree-Based Modelling of Lexical Complexity using Word Embeddings. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Irene Russo. 2021. archer at SemEval-2021 Task 1: Contextualising Lexical Complexity. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. Making it simplext: Implementation and evaluation of a text simplification system for spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):1–36.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Neil Shirude, Sagnik Mukherjee, Tushar Shandhilya, Ananta Mukherjee, and Ashutosh Modi. 2021. IITK@LCP at SemEval-2021 Task 1: Classification for Lexical Complexity Regression Task. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Greta Smolenska, Peter Kolb, Sinan Tang, Mironas Bitinis, Héctor Hernández, and Elin Asklöv. 2021. CLULEX at SemEval-2021 Task 1: A Simple System Goes a Long Way. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Anders Søgaard, Sebastian Ebert, Joost Bastings, and Katja Filippova. 2020. We need to talk about random splits. *arXiv preprint arXiv:2005.00636*.
- Sanjay S.P, Anand Kumar M, and Soman K P. 2016. [AmritaCEN at SemEval-2016 Task 11: Complex Word Identification using Word Embedding](#). In



- Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1022–1027, San Diego, California. Association for Computational Linguistics.
- Regina Stodden and Gayatri Venugopal. 2021. RS.GV at SemEval-2021 Task 1: Sense Relative Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- David Strohmaier, Sian Gooding, Shiva Taslimipoor, and Ekaterina Kochmar. 2020. **SeCoDa: Sense complexity dataset**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5962–5967, Marseille, France. European Language Resources Association.
- Yuki Taya, Lis Kanashiro Pereira, Fei Cheng, and Ichiro Kobayashi. 2021. OCHADAI-KYODAI at SemEval-2021 Task 1: Enhancing Model Generalization and Robustness for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Giuseppe Vettigli and Antonio Sorgente. 2021. CompNA at SemEval-2021 Task 1: Prediction of lexical complexity analyzing heterogeneous features. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Katja Voskoboinik. 2021. katildakat at SemEval-2021 Task 1: Lexical Complexity Prediction of Single Words and Multi-Word Expressions in English. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Nikhil Wani, Sandeep Mathias, Jayashree Aanand Gajjam, and Pushpak Bhattacharyya. 2018. The Whole is Greater than the Sum of its Parts: Towards the Effectiveness of Voting Ensemble Classifiers for Complex Word Identification. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Krzysztof Wróbel. 2016. **PLUJAGH at SemEval-2016 Task 11: Simple System for Complex Word Identification**. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 953–957, San Diego, California. Association for Computational Linguistics.
- Rong Xiang, Jinghang Gu, Emmanuele Chersoni, Wenjie Li, Qin Lu, and Chu-Ren Huang. 2021. PolyU CBS-Comp at SemEval-2021 Task 1: Lexical Complexity Prediction (LCP). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Tuqa Bani Yaseen, Qusai Ismail, Sarah Al-Omari, Eslam Al-Sobh, and Malak Abdullah. 2021. JUSTBLUE at SemEval-2021 Task 1: Predicting Lexical Complexity using BERT and RoBERTa Pre-trained Language Models. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Luci Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. **CWIG3G2 - complex word identification task across three text genres and two user groups**. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Zheng Yuan, Gladys Tyen, and David Strohmaier. 2021. Cambridge at SemEval-2021 Task 1: An Ensemble of Feature-Based and Neural Models for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- George-Eduard Zaharia, Dumitru-Clementin Cercel, and Mihai Dascalu. 2021. UPB at SemEval-2021 Task 1: Combining Deep Learning and Hand-Crafted Features for Lexical Complexity Prediction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. **Complex Word Identification: Challenges in Data Annotation and System Performance**. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. **MacSaar at SemEval-2016 Task 11: Zipfian and Character Features for Complex Word Identification**. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1001–1005, San Diego, California. Association for Computational Linguistics.

## A Full Results

Rank	Team	Pearson	Spearman	MAE	MSE	R2
1	JUST BLUE	0.7886	0.7369	0.0609	0.0062	0.6172
2	DeepBlueAI	0.7882	0.7425	0.0610	0.0061	0.6210
3	Alejandro Mosquera	0.7790	0.7355	0.0619	0.0064	0.6062
4	Andi	0.7782	0.7287	0.0637	0.0064	0.6036
5	CS-UM6P	0.7779	0.7366	0.0803	0.0100	0.3813
6	tuqa	0.7772	0.7344	0.0635	0.0068	0.5771
7	OCHADAI-KYOTO	0.7772	0.7313	0.0617	0.0065	0.6015
8	BigGreen	0.7749	0.7294	0.0629	0.0065	0.5983
9	CSECU-DSG	0.7716	0.7326	0.0632	0.0066	0.5909
10	ia pucp	0.7704	0.7361	0.0618	0.0066	0.5929
11	CLP	0.7692	0.7336	0.0631	0.0067	0.5854
12	ess	0.7656	0.7308	0.0635	0.0069	0.5747
13	ismail2022	0.7653	0.7245	0.0641	0.0069	0.5766
14	andi_gpu	0.7651	0.7275	0.0629	0.0068	0.5810
15	TUDA-CCL	0.7649	0.7164	0.0643	0.0067	0.5846
16	rg_pa	0.7628	0.7251	0.0634	0.0069	0.5749
17	ren_wo_xing	0.7618	0.7229	0.0639	0.0069	0.5715
18	CLULEX	0.7588	0.7089	0.0649	0.0069	0.5753
19	acccb	0.7586	0.7207	0.0635	0.0069	0.5730
20	jiu_mo_zhi	0.7584	0.7175	0.0635	0.0070	0.5691
21	Eslam93	0.7577	0.7224	0.0640	0.0070	0.5648
22	archer	0.7561	0.7067	0.0641	0.0069	0.5707
23	Cambridge	0.7556	0.7105	0.0646	0.0070	0.5705
24	eee	0.7553	0.7203	0.0673	0.0078	0.5181
25	CompNA	0.7552	0.7153	0.0641	0.0070	0.5701
26	LAST	0.7534	0.6988	0.0652	0.0070	0.5652
27	Stanford MLab	0.7533	0.7044	0.0653	0.0071	0.5615
28	mau_lih	0.7513	0.7263	0.0645	0.0071	0.5587
29	IITK@LCP	0.7511	0.7167	0.0654	0.0071	0.5598
30	cognience	0.7510	0.7193	0.0652	0.0071	0.5625
31	qnamqj	0.7509	0.7086	0.0649	0.0072	0.5536
32	feras1515	0.7503	0.7180	0.0652	0.0073	0.5477
33	eslam	0.7482	0.7237	0.0649	0.0072	0.5525
34	RS_GV	0.7478	0.7077	0.0698	0.0079	0.5144
35	LucasHub	0.7434	0.6995	0.0658	0.0073	0.5486
36	LRL_NC	0.7402	0.7013	0.0661	0.0074	0.5440
37	Manchester Metropolitan	0.7389	0.7135	0.0656	0.0074	0.5398
38	UPB	0.7340	0.6785	0.0699	0.0079	0.5098
39	KFU	0.7201	0.6899	0.0687	0.0079	0.5109
40	PolyU CBS-Comp	0.7188	0.6935	0.0682	0.0078	0.5162
41	LCP_RIT	0.7086	0.6535	0.0716	0.0086	0.4695
42	UNBNLP	0.6953	0.6544	0.0716	0.0089	0.4495
43	chenshi	0.6951	0.6532	0.0740	0.0091	0.4366
44	UTFPR	0.6875	0.6588	0.0735	0.0088	0.4577
45	Katildakat	0.6715	0.6454	0.0756	0.0096	0.4060
46	jct	0.6663	0.6457	0.0736	0.0091	0.4402
47	LECCE	0.6452	0.6405	0.0772	0.0096	0.4046
48	S3003183	0.5834	0.5437	0.0804	0.0110	0.3182
–	<i>Frequency Baseline</i>	0.5287	0.5263	0.0870	0.0136	0.2779
49	C3SL	0.4598	0.3983	0.0866	0.0130	0.1989
50	SINAI	0.4428	0.3961	0.0875	0.0131	0.1930
51	ProjectLIN513	0.3884	0.4316	0.1019	0.0159	0.0198
52	glitterosu	0.1807	0.1516	0.1024	0.0194	-0.2016
53	PyGuajo	0.0971	0.1440	0.1166	0.0338	-1.0861
54	RACAI	-0.0272	-0.0268	0.2777	0.1270	-6.8449

Table 4: Sub-task 1: Results and rank (in brackets) in terms of Pearson, Spearman, MAE, MSE, and R2. The rank corresponds to Pearson.

Rank	Team	Pearson	Spearman	MAE	MSE	R2
1	DeepBlueAI	0.8612	0.8526	0.0616	0.0063	0.7389
2	rg_pa	0.8575	0.8529	0.0672	0.0072	0.7035
3	xiang_wen_tian	0.8571	0.8548	0.0675	0.0072	0.7012
4	andi_gpu	0.8543	0.8448	0.0664	0.0071	0.7055
5	ren_wo_xing	0.8541	0.8473	0.0677	0.0073	0.6967
6	Andi	0.8506	0.8381	0.0667	0.0070	0.7107
7	CS-UM6P	0.8489	0.8406	0.0760	0.0087	0.6380
8	OCHADAI-KYOTO	0.8438	0.8285	0.0660	0.0070	0.7103
9	LAST	0.8417	0.8299	0.0677	0.0072	0.7030
10	KFU	0.8406	0.8337	0.0686	0.0073	0.6967
11	jiu_mo_zhi	0.8355	0.8277	0.0710	0.0083	0.6560
12	CSECU-DSG	0.8311	0.8153	0.0678	0.0077	0.6825
13	accceb	0.8310	0.8157	0.0697	0.0076	0.6850
14	Stanford MLab	0.8280	0.8124	0.0711	0.0080	0.6671
15	IITK@LCP	0.8277	0.8228	0.0811	0.0098	0.5949
16	qnamqj	0.8246	0.8227	0.0787	0.0094	0.6097
17	LRL_NC	0.8244	0.8156	0.0702	0.0079	0.6737
18	mau_lih	0.8234	0.8211	0.0790	0.0096	0.6042
19	TUDA-CCL	0.8190	0.8091	0.0711	0.0080	0.6677
20	Alejandro Mosquera	0.8093	0.8017	0.0731	0.0084	0.6519
21	LucasHub	0.8000	0.7797	0.0754	0.0089	0.6323
22	UPB	0.7962	0.7988	0.0788	0.0099	0.5917
23	CompNA	0.7931	0.7800	0.0783	0.0093	0.6160
24	justglowing	0.7902	0.7851	0.0786	0.0092	0.6169
25	BigGreen	0.7898	0.7769	0.0903	0.0124	0.4858
26	Katildakat	0.7848	0.7869	0.0807	0.0101	0.5816
27	Manchester Metropolitan	0.7611	0.7711	0.0806	0.0102	0.5770
28	UTFPR	0.7601	0.7504	0.0817	0.0102	0.5754
29	UNBNLP	0.7515	0.7420	0.0802	0.0106	0.5623
30	chenshi	0.7500	0.7497	0.0867	0.0112	0.5365
31	PolyU CBS-Comp	0.7416	0.7222	0.0839	0.0109	0.5473
32	cognience	0.7232	0.7301	0.0851	0.0117	0.5144
–	<i>Frequency Baseline</i>	0.6571	0.6345	0.0924	0.0140	0.4030
33	C3SL	0.3941	0.3675	0.1145	0.0206	0.1470
34	PyGuajo	0.3931	0.3902	0.1132	0.0205	0.1488
35	SINAI	0.3197	0.3508	0.1217	0.0243	-0.0062
36	LECCE	0.2821	0.3138	0.1202	0.0226	0.0624
37	glitterosu	0.1860	0.1316	0.1332	0.0255	-0.0564

Table 5: Sub-task 2: Results and rank (in brackets) in terms of Pearson, Spearman, MAE, MSE, and R2. The Rank Corresponds to Pearson.

## B System Features

Team	Features	Classification Approach	System Paper
Alejandro Mosquera	Length, Frequency, Semantic, Sentence	Gradient Boosted Regression	(Mosquera, 2021)
Andi	Psycholinguistic, Glove, Word2Vec, ConceptNet NumberBatch, BERT, RoBERTa, ELECTRA, ALBERT, DeBERTa	Ridge Regression, Gradient Boosted Regression	(Rotaru, 2021)
Archer	Length, Frequency, Psycholinguistic, Scrabble Score, Word Inclusion, Semantic	Random Forest Regression, Gradient Boosted Regression	(Russo, 2021)
BigGreen	Length, Semantic, Glove, Elmo, InferSent, Phonetic, Frequency, POS	Gradient Boosted Regression, BERT	(Islam et al., 2021)
C3SL	Sent2Vec	Multi-layer Perceptron	
Cambridge	Frequency, Syntactic, Length	BERT, Random Forest Regression	(Yuan et al., 2021)
CLULEX	Frequency, POS, Named Entities, Word Inclusion, Sentence, Bert	Decision Tree	(Smolenska et al., 2021)
CompNA	Length, Semantic, Glove, Word Inclusion, Token and Context Encoded	Decision Tree Ensemble	(Vettigli and Sorgente, 2021)
CS-UM6P	Token and Context Encoded	BERT, RoBERTa	(Mamoun et al., 2021)
CSECU-DSG	Token and Context Encoded	BERT, RoBERTa	(Aziz et al., 2021)
DeepBlueAI	Token and Context Encoded	BERT, ALBERT, RoBERTa, ERNIE	(Pan et al., 2021)
Hub	TF-IDF, Context Encoded	RoBERTa, Inception	(Huang et al., 2021)
IA PUCP	Sentence, POS, N-gram Frequency, RoBERTa, XLNet, BERT	Gradient Boosted Regression	(Rojas and Alva-Manchego, 2021)
IITK@LCP	Electra + Glove	Linear Regression, Support Vector Machine	(Shirude et al., 2021)
JCT	POS, Frequency, BERT, Cluster Features	Gradient Boosted Regression	(Liebeskind et al., 2021)
JUST BLUE	Token Encoded and Context Encoded	Average of Weighted Bert and Roberta	(Yaseen et al., 2021)
Katildakat	BERT, Length, BERT-score, Frequency, Semantic,	Linear Regression, Multi-layer Perceptron	(Voskoboinik, 2021)
LAST	Frequency, Psycholinguistic, Sentence, Bigram Association	Gradient Boosted Regression	(Bestgen, 2021)
LCP-RIT	Length, Frequency, Character N-Grams, Psycholinguistic, POS	Random forest Regressor	(Desai et al., 2021)
LRL_NC	Frequency, Semantic, Laanguage Model, Psycholinguistic, Word Inclusion	Random forest regressor	
Manchester Metropolitan	Frequency, Psycholinguistic, Length, Embeddings	CNN	(Flynn and Shardlow, 2021)
OCHADAI-KYOTO	Token and Context Encoded	BERT, RoBERTa	(Taya et al., 2021)
PolyU CBS-Comp	Frequency, Length, Capitalisation, POS, Embeddings, BERT, GPT-2	Gradient Boosted Regression	(Xiang et al., 2021)
RG_PA	Context Encoded	RoBERTa	(Rao et al., 2021)
RS.GV	GLoVe, ELMo, BERT, Flair, Readability, Length, Frequency, Semantic, Psycholinguistic, Morphological, Word Inclusion, Named Entity	Feed-Forward Neural Network	(Stodden and Venugopal, 2021)
Stanford MLab	Glove, Length, POS, Named Entity	Gradient Boosted Regression	(Rozi et al., 2021)
TUDA-CCL	Linguistic, Semantic, Embeddings, Psycholinguistic, Frequencies, Word Inclusion	Gradient Boosted Regression	(Gombert and Bartsch, 2021)
UNBNLP	Length, Frequency, Character-Level-Encoder, BERT	Neural Network, Support Vector Machine	(King et al., 2021)
UPB	Transformers, Word Embeddings, Character Embeddings, Length, Psycholinguistic	BERT, RoBERTa, Regression	(Zaharia et al., 2021)
UTFPR	Frequency, Length, Semantic, Bert Embedding	Support Vector Machine	(Paetzold, 2021)

Table 6: Systems that participated and submitted a paper, the features and classification approaches they employed.

# OCHADAI-KYOTO at SemEval-2021 Task 1: Enhancing Model Generalization and Robustness for Lexical Complexity Prediction

Yuki Taya<sup>1</sup>, Lis Kanashiro Pereira<sup>1</sup>, Fei Cheng<sup>2</sup>, and Ichiro Kobayashi<sup>1</sup>

<sup>1</sup>Ochanomizu University, Japan

<sup>2</sup>Kyoto University, Japan

g1620525@is.ocha.ac.jp, kanashiro.pereira@ocha.ac.jp,

feicheng@i.kyoto-u.ac.jp, koba@is.ocha.ac.jp

## Abstract

We propose an ensemble model for predicting the lexical complexity of words and multiword expressions (MWEs). The model receives as input a sentence with a target word or MWE and outputs its complexity score. Given that a key challenge with this task is the limited size of annotated data, our model relies on pretrained contextual representations from different state-of-the-art transformer-based language models (i.e., BERT and RoBERTa), and on a variety of training methods for further enhancing model generalization and robustness: multi-step fine-tuning and multi-task learning, and adversarial training. Additionally, we propose to enrich contextual representations by adding hand-crafted features during training. Our model achieved competitive results and ranked among the top-10 systems in both subtasks.

## 1 Introduction

Predicting the difficulty of a word in a given context is useful in many natural language processing (NLP) applications such as lexical simplification. Previous efforts (Paetzold and Specia, 2016; Yimam et al., 2018; Zampieri et al., 2017) have focused on framing this as a binary classification task, which might not be ideal, since a word close to the decision boundary is assumed to be just as complex as one further away (Shardlow et al., 2020). To alleviate this issue, SemEval-2021 Task 1 (Shardlow et al., 2021a) formulates this task as a regression task, where a model should predict the complexity value of words (Subtask 1) and MWEs (Subtask 2) in context.

This paper describes the system developed by the Ochadai-Kyoto team for SemEval-2021 Task 1. Given that a key challenge in this task is the limited size of annotated data, we follow best practices from recent work on enhancing model generalization and robustness, and propose a model

Task	Domain	Train	Trial	Test
Subtask 1 (single-word)	Europarl	2512	143	345
	Biomed	2576	135	289
	Bible	2574	143	283
	All	7662	421	917
Subtask 2 (MWE)	Europarl	498	37	65
	Biomed	514	33	53
	Bible	505	29	66
	All	1517	99	184

Table 1: Summary of the Complex dataset.

ensemble that leverages pretrained representations (i.e. BERT and RoBERTa), multi-step fine-tuning, multi-task learning and adversarial training. Additionally, we propose to enrich contextual representations by incorporating hand-crafted features during training. Our model ranked 7th out of 54 participating teams on Subtask 1, and 8th out of 37 teams on Subtask 2, obtaining Pearson correlation scores of 0.7772 and 0.8438, respectively.

## 2 Task Description

SemEval-2021 Task 1 provides participants with an augmented version of the CompLex dataset (Shardlow et al., 2020), a multi-domain English dataset with sentences containing words and MWEs annotated on a continuum scale of complexity, in the range of [0,1]. Easier words and MWEs are assigned lower complexity scores, while the more challenging ones are assigned higher scores. This corpus contains a balanced number of sentences from three different domains: Bible (Christodouloupoulos and Steedman, 2015), Europarl (Koehn, 2005) and Biomedical (Bada et al., 2012). The task is to predict the complexity value of single words (Subtask 1) and MWEs (Subtask 2) in context. The statistics of the corpus are presented in Table 1. Our team participated in both subtasks, and the next section outlines the overview of our model.

### 3 System Overview

We focus on exploring different training techniques using BERT and RoBERTa, given their superior performance on a wide range of NLP tasks. Each text encoder and training method used in our model are detailed below.

#### 3.1 Text Encoders

**BERT** (Devlin et al., 2019): We use the BERT<sub>BASE</sub> model released by the authors. It consists of 12 transformer layers, 12 self-attention heads per layer, and a hidden size of 768.

**RoBERTa** (Liu et al., 2019b): We use both the RoBERTa<sub>BASE</sub> and RoBERTa<sub>LARGE</sub> models released by the authors. Similar to BERT, RoBERTa<sub>BASE</sub> consists of 12 transformer layers, 12 self-attention heads per layer, and a hidden size of 768. RoBERTa<sub>LARGE</sub> consists of 24 transformer layers, 16 self-attention heads per layer, and a hidden size of 1024.

#### 3.2 Training Procedures

**Standard fine-tuning:** This is the standard fine-tuning procedure where we fine-tune BERT and RoBERTa on each subtask-specific data.

**Feature-enriched fine-tuning (FEAT):** During training, we enrich BERT and RoBERTa representations with word frequency information of the target word or MWE. We compute the log frequency values using the Wiki40B corpus (Guo et al., 2020). For MWEs, we compute the log of the average of the frequency of each component word. After applying the min-max normalization to this feature, we concatenate it to the CLS token vector obtained from the last layer of BERT and RoBERTa.

**Multi-step fine-tuning (MSFT):** Multi-step fine-tuning works by performing a second stage of pre-training with data-rich related supervised tasks. It has been shown to improve model robustness and performance, especially for data-constrained scenarios (Phang et al., 2018; Camburu et al., 2019). Due to the limited size of the data provided for Subtask 2, we first fine-tune BERT and RoBERTa on the Subtask 1 dataset. This model’s parameters are further refined by fine-tuning on the Subtask 2 dataset.

**Multi-task learning (MTL):** Multi-task learning is an effective training paradigm to promote model generalization ability and performance (Caruana, 1997; Liu et al., 2015, 2019a; Ruder, 2017; Collobert et al., 2011). It works by leveraging data

from many (related) tasks. In our experiments, we use the MT-DNN framework (Liu et al., 2019a, 2020b), which incorporates BERT and RoBERTa as the shared text encoding layers (shared across all tasks), while the top layers are task-specific. We used the pre-trained BERT and RoBERTa models to initialize its shared layers and refined them via MTL on both subtasks (i.e. Subtask 1 and Subtask 2).

**Adversarial training (ADV):** Adversarial training has proven effective in improving model generalization and robustness in computer vision (Madry et al., 2017; Goodfellow et al., 2014) and more recently in NLP (Zhu et al., 2019; Jiang et al., 2019; Cheng et al., 2019; Liu et al., 2020a; Pereira et al., 2020). It works by augmenting the input with a small perturbation that maximizes the adversarial loss:

$$\min_{\theta} \mathbb{E}_{(x,y) \sim D} [\max_{\delta} l(f(x + \delta; \theta), y)] \quad (1)$$

where the inner maximization can be solved by projected gradient descent (Madry et al., 2017). Recently, adversarial training has been successfully applied to NLP as well (Zhu et al., 2019; Jiang et al., 2019; Pereira et al., 2020). In our experiments, we use SMART (Jiang et al., 2019), which instead regularizes the standard training objective using *virtual adversarial training* (Miyato et al., 2018):

$$\min_{\theta} \mathbb{E}_{(x,y) \sim D} [l(f(x; \theta), y) + \alpha \max_{\delta} l(f(x + \delta; \theta), f(x; \theta))] \quad (2)$$

Effectively, the adversarial term encourages smoothness in the input neighborhood, and  $\alpha$  is a hyperparameter that controls the trade-off between standard errors and adversarial errors.

#### 3.3 Ensemble Model

Ensemble of deep learning models has proven effective in improving test accuracy (Allen-Zhu and Li, 2020). We built different ensemble models by taking an unweighted average of the outputs of a few independently trained models. Each single model was trained on standard fine-tuning, multi-step fine-tuning, multi-task learning, or adversarial training, using different text encoders (i.e. BERT or RoBERTa).

## 4 Experiments

### 4.1 Implementation Details

Our model implementation is based on the MT-DNN framework (Liu et al., 2019a, 2020b). We

use BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019b) as the text encoders. We used ADAM (Kingma and Ba, 2015) as our optimizer with a learning rate in the range  $\in \{8 \times 10^{-6}, 9 \times 10^{-6}, 1 \times 10^{-5}\}$  and a batch size  $\in \{8, 16, 32\}$ . The maximum number of epochs was set to 10. A linear learning rate decay schedule with warm-up over 0.1 was used, unless stated otherwise. To avoid gradient exploding, we clipped the gradient norm within 1. All the texts were tokenized using wordpieces and were chopped to spans no longer than 512 tokens. During adversarial training, we follow (Jiang et al., 2019) and set the perturbation size to  $1 \times 10^{-5}$ , the step size to  $1 \times 10^{-3}$ , and to  $1 \times 10^{-5}$  the variance for initializing the perturbation. The number of projected gradient steps and the  $\alpha$  parameter (Equation 2) were both set to 1.

We follow (Devlin et al., 2019), and set the first token as the [CLS] token when encoding the input. For Subtask 1, we separate the input sentence and the target token with the special token [SEP]. e.g. [CLS] This was the *length* of Sarah’s life [SEP] *length* [SEP]. For Subtask 2, such encoding led to lower performance of our system. Therefore, we consider only the target MWE when encoding the input, e.g. [CLS] *financial world* [SEP].

For each subtask, we used the trial dataset released by organizers as development set (see Table 1). We select the best epoch and the best hyper-parameters using performance (measured in terms of Pearson correlation score) on this development set. We also experimented on saving the best epoch and best hyper-parameters for each domain (Bible, Biomedical and Europarl).

## 4.2 Main Results

Submitted systems were evaluated on five metrics: Pearson correlation (R), Spearman correlation (Rho), Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R2). The systems were ranked from highest Pearson correlation score to lowest. We built several models that use different text encoders and different training methods, as described in Section 3. See Table 2 for the results. First, we observe that ensembling different single models yield better performance on both tasks. Furthermore, models that use feature-enriched representations, multi-task learning, multi-step fine-tuning and adversarial training surpass models that use the standard fine-tuning approach. We detail next the results for each subtask.

For Subtask 1, the single models that used RoBERTa, adversarial training, multi-task learning and feature-enriched representations performed best on the development set. Moreover, saving the best epoch and hyper-parameters for each domain performed better than saving the best epoch and hyper-parameters without domain distinction. Among the single models, the model that performed best on the development set was the model that uses RoBERTa<sub>LARGE</sub> and adversarial training (RoBERTa<sub>LARGE</sub>(ADV)<sub>domain</sub> model, with a Pearson score of 0.8441). The second best single model was the model that uses RoBERTa<sub>BASE</sub> and feature-enriched contextual representations (RoBERTa<sub>BASE</sub>(FEAT)<sub>domain</sub> model, with a Pearson score of 0.8391). The third best single model was the model that uses RoBERTa<sub>LARGE</sub> and multi-task learning (RoBERTa<sub>LARGE</sub>(MTL)<sub>domain</sub> model, with a Pearson score of 0.8371). Thus, we ensemble these three single models in different ways when making our submissions. The ensemble model that performed best on the test set (Ensemble  $2_{\text{single\_word}}$ ) was the model that combined feature-enriched contextual representations (RoBERTa<sub>BASE</sub>(FEAT)<sub>domain</sub>), adversarial training (RoBERTa<sub>LARGE</sub>(ADV)<sub>domain</sub>), and multi-task learning (RoBERTa<sub>LARGE</sub>(MTL)<sub>domain</sub>). This ensemble model obtained development and test set Pearson scores of 0.8570 and 0.7772, respectively.

For Subtask 2, the single models that use BERT<sub>BASE</sub> outperformed models that use RoBERTa, on the development set. Moreover, we noted that using the Subtask 1 dataset as auxiliary dataset by performing multi-step fine-tuning and multi-task learning greatly help to improve the performance. For instance, the BERT<sub>BASE</sub>(MSFT) outperformed the BERT<sub>BASE</sub> model by 0.0405 Pearson correlation points (0.7965 vs 0.8370). The ensemble model that performed best on the test set (Ensemble  $1_{\text{MWE}}$ ) was the model that combined multi-step fine-tuning and multi-task learning using BERT, i.e. BERT<sub>BASE</sub>(MSFT) and BERT<sub>BASE</sub>(MTL) models, respectively, and multi-task learning using RoBERTa (RoBERTa<sub>LARGE</sub>(MTL) model). This ensemble model obtained development and test set Pearson scores of 0.8461 and 0.8438, respectively. Different from Subtask 1, we observe that saving the best epoch and hyper-parameters for each domain on the development set performed worse than saving the best epoch and hyper-parameters without do-

Training Methods	Ensemble			R	Rho	MAE	MSE	R2
<b>Subtask 1 (Single Word Lexical Complexity Prediction Task)</b>								
BERT <sub>BASE</sub> <sup>dev</sup>				0.7794	0.7423	0.0664	0.0077	0.1898
RoBERTa <sub>BASE</sub> <sup>dev</sup>				0.8139	0.7498	0.0628	0.0064	0.4325
RoBERTa <sub>BASE</sub> (FEAT) <sup>dev</sup>	✓			0.8348	0.7579	0.0603	0.0058	0.6955
RoBERTa <sub>BASE</sub> (FEAT) <sup>dev</sup> <sub>domain</sub>		✓	✓	0.8391	0.7640	0.0599	0.0057	0.6976
RoBERTa <sub>LARGE</sub> <sup>dev</sup>				0.8213	0.7629	0.0627	0.0062	0.5381
RoBERTa <sub>LARGE</sub> (FEAT) <sup>dev</sup> <sub>domain</sub>				0.8218	0.7513	0.0634	0.0063	0.6025
RoBERTa <sub>LARGE</sub> (MTL) <sup>dev</sup> <sub>domain</sub>		✓	✓	0.8371	0.7694	0.0609	0.0062	0.3640
RoBERTa <sub>LARGE</sub> (ADV) <sup>dev</sup>	✓			0.8328	0.7760	0.0603	0.0059	0.5509
RoBERTa <sub>LARGE</sub> (ADV) <sup>dev</sup> <sub>domain</sub>		✓		<b>0.8441</b>	<b>0.7873</b>	<b>0.0572</b>	<b>0.0054</b>	<b>0.7123</b>
Ensemble 1 <sub>single_word</sub> <sup>dev</sup>	○			0.8481	0.7825	0.0578	0.0053	0.7175
Ensemble 2 <sub>single_word</sub> <sup>dev</sup>		○		<b>0.8570</b>	<b>0.7902</b>	<b>0.0553</b>	<b>0.0050</b>	<b>0.7335</b>
Ensemble 3 <sub>single_word</sub> <sup>dev</sup>			○	0.8548	0.7816	0.0560	0.0051	0.7300
Ensemble 1 <sub>single_word</sub> <sup>test</sup>	○			0.7590	0.7174	0.0640	0.0069	0.5719
Ensemble 2 <sub>single_word</sub> <sup>test</sup>		○		<b>0.7772</b>	<b>0.7313</b>	<b>0.0617</b>	<b>0.0065</b>	<b>0.6015</b>
Ensemble 3 <sub>single_word</sub> <sup>test</sup>			○	0.7761	0.7244	0.0622	0.0065	0.6003
Top Team Result (JUST BLUE) <sub>single_word</sub> <sup>test*</sup>				<b>0.7886</b>	<b>0.7369</b>	<b>0.0609</b>	<b>0.0062</b>	<b>0.6172</b>
<b>Subtask 2 (MWE Lexical Complexity Prediction Task)</b>								
BERT <sub>BASE</sub> (full context) <sup>dev †</sup>				0.7903	0.7839	0.0770	0.0090	0.6240
BERT <sub>BASE</sub> <sup>dev</sup>				0.7965	0.7856	0.0761	0.0086	0.3552
BERT <sub>BASE</sub> (FEAT) <sup>dev</sup>				0.8166	0.8033	0.0730	0.0080	0.6610
BERT <sub>BASE</sub> (MSFT) <sup>dev</sup>	✓			0.8370	0.8361	<b>0.0661</b>	0.0071	0.5276
BERT <sub>BASE</sub> (MSFT) <sup>dev</sup> <sub>domain</sub>		✓	✓	<b>0.8498</b>	<b>0.8492</b>	0.0669	0.0068	0.7099
BERT <sub>BASE</sub> (MTL) <sup>dev</sup>	✓			0.8176	0.8202	0.0725	0.0081	0.5086
BERT <sub>BASE</sub> (MTL) <sup>dev</sup> <sub>domain</sub>		✓	✓	0.8442	0.8323	0.0667	<b>0.0067</b>	<b>0.7125</b>
RoBERTa <sub>BASE</sub> <sup>dev</sup>				0.7689	0.7659	0.0771	0.0098	0.3767
RoBERTa <sub>LARGE</sub> <sup>dev</sup>				0.8110	0.8181	0.0737	0.0082	0.4363
RoBERTa <sub>LARGE</sub> (MTL) <sup>dev</sup>	✓			0.8176	0.8202	0.0725	0.0081	0.5086
RoBERTa <sub>LARGE</sub> (MTL) <sup>dev</sup> <sub>domain</sub>			✓	0.8341	0.8276	0.0675	0.0075	0.6790
RoBERTa <sub>LARGE</sub> (ADV) <sup>dev</sup>				0.8119	0.8019	0.0718	0.0080	0.4785
RoBERTa <sub>LARGE</sub> (ADV&MSFT) <sup>dev</sup>				0.8247	0.8092	0.0685	0.0076	0.4748
RoBERTa <sub>LARGE</sub> (ADV&MSFT) <sup>dev</sup> <sub>domain</sub>		✓		0.8283	0.8176	0.0676	0.0074	0.6858
Ensemble 1 <sub>MWE</sub> <sup>dev</sup>	○			0.8461	0.8441	0.0672	0.0068	0.7080
Ensemble 2 <sub>MWE</sub> <sup>dev</sup>		○		0.8543	0.8444	0.0642	<b>0.0064</b>	<b>0.7270</b>
Ensemble 3 <sub>MWE</sub> <sup>dev</sup>			○	<b>0.8571</b>	<b>0.8509</b>	<b>0.0640</b>	<b>0.0064</b>	0.7267
Ensemble 1 <sub>MWE</sub> <sup>test</sup>	○			<b>0.8438</b>	<b>0.8285</b>	<b>0.0660</b>	<b>0.0070</b>	<b>0.7103</b>
Ensemble 2 <sub>MWE</sub> <sup>test</sup>		○		0.8376	0.8231	0.0682	0.0076	0.6840
Ensemble 3 <sub>MWE</sub> <sup>test</sup>			○	0.8312	0.8157	0.0708	0.0080	0.6686
Top Team Result (DeepBlueAI) <sub>single_word</sub> <sup>test*</sup>				<b>0.8612</b>	<b>0.8526</b>	<b>0.0616</b>	<b>0.0063</b>	<b>0.7389</b>

Table 2: Comparison of different text encoders and different training methods on the single word lexical complexity prediction task (Subtask 1) and on the MWE lexical complexity prediction task (Subtask 2). Best results for single and ensemble models are highlighted in **bold**. † indicates that we consider the full context surrounding the MWE when encoding the input. In the other models for Subtask 2, we consider only the target MWE. \* indicates results obtained from the Task’s official leaderboard: (<https://competitions.codalab.org/competitions/27420#results>). ✓ indicates each single model that was used in the ensemble, indicated in each column by ○.

main distinction. We hypothesize that, due to the small size of the data provided for Subtask 2, saving the best epoch and hyper-parameters without domain distinction might avoid overfitting.

## 5 Analysis

We briefly analyse our best models’ results on the test set for each subtask. Figure 1 (top) shows a comparison between our best ensemble model’s

predictions for Subtask 1 (Ensemble 2<sub>single\_word</sub>) and the gold answers. We observe that our model often fails to predict correctly in the range where samples have a complexity score below 0.2. We hypothesize this might be due to the skewed distribution of the golden complexity scores for each domain, as shown in Table 4. A possible solution might be to build domain-specific models.

Figure 1 (bottom) shows a comparison between the best ensemble model’s predictions for Subtask



Domain	Sentence	Target	Prediction	Label
<b>Sub-task 1</b>				
Europarl	The Swedish Presidency aims to maintain the debate on animal welfare and good animal <i>husbandry</i> .	<i>husbandry</i>	0.3270	0.53143
Biomed	We adopted the same strategy to investigate the relative contribution of the 129 <i>Chromosome 1</i> segment and the Apes gene to each disease trait.	<i>Chromosome</i>	0.4865	0.2237
Bible	God has gone up with a <i>shout</i> , Yahweh with the sound of a trumpet.	<i>shout</i>	<b>0.2032</b>	<b>0.2031</b>
<b>Sub-task 2</b>				
Biomed	These studies strongly suggest that the hsp family of proteins has <i>other functions</i> in addition to protecting proteins and cells during stress.	<i>other functions</i>	0.2564	0.4167
Europarl	What plans does the Commission have to introduce <i>eco labelling</i> of 'sustainable' palm oils?	<i>eco labelling</i>	0.5277	0.3553
Bible	In the <i>dry season</i> , they vanish.	<i>dry season</i>	<b>0.2832</b>	<b>0.2857</b>

Table 3: Examples of successful and poor predictions on the test set by the best ensemble models submitted for each subtask (Ensemble  $2_{\text{single\_word}}$  and Ensemble  $1_{\text{MWE}}$  models). Successful predictions are highlighted in **bold**.

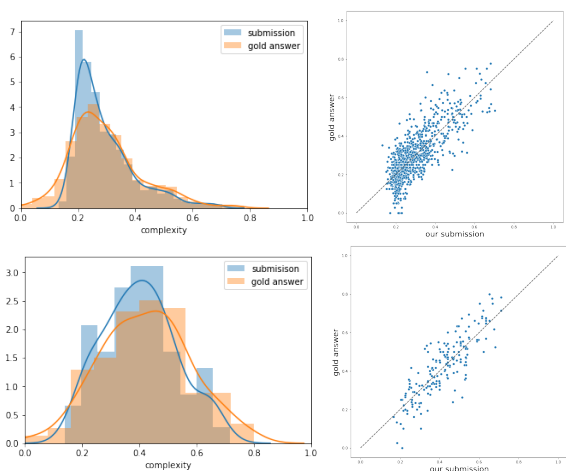


Figure 1: Comparison between the Ensemble  $2_{\text{single\_word}}$  and Ensemble  $1_{\text{MWE}}$  models' predictions submitted for Sub-task 1 (top) and Sub-task 2 (bottom), respectively, and the gold answers. On the left, we show the distribution of the correct complexity score and our submission. On the right, we show a scatter plot where the x-axis corresponds to our model's predictions and the y-axis corresponds to the gold answers.

2 (Ensemble  $1_{\text{MWE}}$ ), and the gold answers. Compared to Subtask 1, the data distribution of the development and test sets of Subtask 2 look more similar, hence a possible reason why the development and test set scores were closer than in Subtask 1 (the best ensemble models obtained development and test set scores of 0.8570 and 0.7772, respectively, in Subtask 1, and 0.8461 and 0.8438, respectively, in Subtask 2). Table 3 shows examples of successful and poor predictions made by Ensemble  $2_{\text{single\_word}}$  and Ensemble  $1_{\text{MWE}}$  models. Table 4 shows how the performance of these models varies across domains. The Biomedical domain obtained the highest Pearson correlation scores on both subtasks, which indicates that

	Bible	Europarl	Biomed
<b>Sub-task 1</b>			
<b>MAE</b>	0.0679	0.0549	0.0638
<b>R</b>	0.7329	0.7213	0.8358
<b>Sub-task 2</b>			
<b>MAE</b>	0.0721	0.0592	0.0667
<b>R</b>	0.8114	0.6374	0.9104

Table 4: Performance of Ensemble  $2_{\text{single\_word}}$  and Ensemble  $1_{\text{MWE}}$  models on each domain and subtask.

might be a sharper difference between simple and complex words in this corpus (Shardlow et al., 2021b).

## 6 Conclusion

In this paper, we have presented the implementation of the Ochadai-Kyoto system submitted to the SemEval-2021 Task 1. Our model ranked 7th out of 54 participating teams on Subtask 1, and 8th out of 37 teams on Subtask 2. We proposed an ensemble model that leverages pretrained representations, multi-step fine-tuning, multi-task learning and adversarial training. We also proposed to enrich contextual representations by incorporating hand-crafted features during training. In future efforts, we plan to further improve our model to better handle data-constraint and domain-shift scenarios.

## References

- Zeyuan Allen-Zhu and Yuanzhi Li. 2020. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*.
- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):1–20.
- Oana-Maria Camburu, Vid Kocijan, Thomas Lukasiewicz, and Yordan Yordanov. 2019. A surprisingly robust trick for the winograd schema challenge.
- Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.
- Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. [Robust neural machine translation with doubly adversarial inputs](#).
- Christos Christodoulopoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.
- Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. 2020. Wiki-40b: Multilingual language model dataset. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2440–2452.
- Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.
- Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *ICLR (Poster) 2015*.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.
- Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020a. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*.
- Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval.
- Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.
- Xiaodong Liu, Yu Wang, Jianshu Ji, Hao Cheng, Xueyun Zhu, Emmanuel Awa, Pengcheng He, Weizhu Chen, Hoifung Poon, Guihong Cao, and Jianfeng Gao. 2020b. The microsoft toolkit of multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:2002.07972*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Lis Pereira, Xiaodong Liu, Fei Cheng, Masayuki Asahara, and Ichiro Kobayashi. 2020. Adversarial training for commonsense inference. *arXiv preprint arXiv:2005.08156*.
- Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.
- Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [Complex: A new corpus for lexical complexity prediction from likert scale data](#).

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. *arXiv preprint arXiv:1710.04989*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Thomas Goldstein, and Jingjing Liu. 2019. FreeLB: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*.

# SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC)

Federico Martelli, Najla Kalach, Gabriele Tola, Roberto Navigli

Sapienza NLP Group

Department of Computer Science

Sapienza University of Rome, Italy

first.lastname@uniroma1.it

## Abstract

In this paper, we introduce the first SemEval task on Multilingual and Cross-Lingual Word-in-Context disambiguation (MCL-WiC). This task allows the largely under-investigated inherent ability of systems to discriminate between word senses within and across languages to be evaluated, dropping the requirement of a fixed sense inventory. Framed as a binary classification, our task is divided into two parts. In the multilingual sub-task, participating systems are required to determine whether two target words, each occurring in a different context within the same language, express the same meaning or not. Instead, in the cross-lingual part, systems are asked to perform the task in a cross-lingual scenario, in which the two target words and their corresponding contexts are provided in two different languages. We illustrate our task, as well as the construction of our manually-created dataset including five languages, namely Arabic, Chinese, English, French and Russian, and the results of the participating systems. Datasets and results are available at: <https://github.com/SapienzaNLP/mcl-wic>.

## 1 Introduction

During recent decades, the field of Natural Language Processing (NLP) has witnessed the development of an increasing number of neural approaches to representing words and their meanings. Word embeddings encode a target word type with one single vector based on co-occurrence information. However, word embeddings conflate different meanings of a single target word into the same representation, thus they fail to capture the polysemous nature of words. To address this limitation, more sophisticated representations such as multi-prototype and contextualized embeddings have been put forward. Multi-prototype embeddings concentrate on the semantics which underlie

a target word by clustering occurrences based on their context similarities (Neelakantan et al., 2015; Pelevina et al., 2016). In an effort to exploit the knowledge derived from lexical-knowledge bases, Iacobacci et al. (2015) introduced a new approach which allows sense representations to be linked to a predefined sense inventory. More recently, contextualized embeddings were proposed. These representations are obtained by means of neural language modeling, e.g. using LSTMs (Melamud et al., 2016) or the Transformer architecture (Devlin et al., 2019; Conneau et al., 2020), and are capable of representing words based on the context in which they occur. Contextualized representations have also been used to obtain effective sense embeddings (Loureiro and Jorge, 2019; Scarlini et al., 2020a,b; Calabrese et al., 2020).

Although virtually all the above approaches can be evaluated in downstream applications, the inherent ability of the various embeddings to capture meaning distinctions still remains largely under-investigated. While Word Sense Disambiguation (WSD), i.e. the task of determining the meaning of a word in a given context (Navigli, 2009), has long explored the aforementioned ability, the task does not make it easy to test approaches that are not explicitly linked to existing sense inventories, such as WordNet (Miller et al., 1990) and BabelNet (Navigli and Ponzetto, 2010). This has two major drawbacks. First, sense inventories are not always available, especially for rare languages. Second, such requirement limits the evaluation of word and sense representations which are not bound to a sense inventory. To tackle this limitation, some benchmarks have recently been proposed. The CoSimLex dataset (Armendariz et al.) and the related SemEval-2020 Task 3 (Armendariz et al., 2020) focus on evaluating the similarity of word pairs which occur in the same context. More recently, the Word-in-Context (WiC) task (Pilehvar

and Camacho-Collados, 2019), included in the SuperGLUE benchmark for Natural Language Understanding (NLU) systems (Wang et al., 2019) and its multilingual extension XL-WiC (Raganato et al., 2020), require systems to determine whether a word occurring in two different sentences is used with the same meaning, without relying on a pre-defined sense inventory. For instance, given the following sentence pair:

- the *mouse* eats the cheese,
- click the right *mouse* button,

the ideal system should establish that the target word *mouse* is used with two different meanings.

Despite the steps forward made in this promising research direction, existing benchmarks suffer from the following shortcomings: i) they are mostly automatically retrieved; ii) they do not enable cross-lingual evaluation scenarios in which systems are tested in different languages at the same time; iii) they do not cover all open-class parts of speech.

In order to address the aforementioned drawbacks, we propose the first SemEval task on Multilingual and Cross-Lingual Word-in-Context disambiguation (MCL-WiC) and present the first entirely manually-annotated dataset for the task. Importantly, MCL-WiC enables new cross-lingual evaluation scenarios covering all open-class parts of speech, as well as a wide range of domains and genres. The dataset is available in five European and non-European languages, i.e. Arabic (Ar), Chinese (Zh), English (En), French (Fr) and Russian (Ru).

## 2 Related Work

Several different tasks have been put forward which go beyond traditional WSD and drop the requirement of fixed sense inventories. Among the first alternatives we cite monolingual and cross-lingual Lexical Substitution (McCarthy and Navigli, 2007; Mihalcea et al., 2010). Word-in-context similarity has also been proposed as a way to capture the dynamic nature of word meanings: the Stanford Contextual Word Similarities (SCWS) dataset, proposed by Huang et al. (2012), contains human judgements on pairs of words in context. Along these same lines, Armendariz et al. introduced CoSimLex, a dataset designed to evaluate the ability of models to capture word similarity judgements provided by humans.

MCL-WiC				
Sub-task	Dataset	Train	Dev	Test
Multilingual	Ar-Ar	-	500	500
	En-En	4000	500	500
	Fr-Fr	-	500	500
	Ru-Ru	-	500	500
	Zh-Zh	-	500	500
Cross-lingual	En-Ar	-	-	500
	En-Fr	-	-	500
	En-Ru	-	-	500
	En-Zh	-	-	500

Table 1: The MCL-WiC dataset: number of unique lexemes divided by sub-task and dataset. The second column (Dataset) indicates the available language combination.

More recently, Pilehvar and Camacho-Collados (2019) presented the Word-in-Context (WiC) dataset. Framed as a binary classification task, WiC is a benchmark for the evaluation of context-dependent embeddings. However, WiC covers only one language, i.e. English, and two parts of speech, namely nouns and verbs. To enable evaluation in languages other than English, Raganato et al. (2020) proposed XL-WiC, an extension of the WiC dataset which covers different European and non-European languages, thus allowing for zero-shot settings. Despite their effectiveness, both the WiC and XL-WiC datasets are not manually created and do not cover all open-class parts of speech. Moreover, they do not consider cross-lingual evaluation scenarios in which systems are tested in more than one language at the same time, thus highlighting the need for a new evaluation benchmark.

## 3 The Multilingual and Cross-lingual Word-in-Context Task

In this Section, we present our SemEval task and describe a new dataset called Multilingual and Cross-lingual Word-in-Context (MCL-WiC). The task is divided into a multilingual and a cross-lingual sub-task, each containing different datasets divided according to language combination. Each dataset instance is focused on a given lexeme<sup>1</sup> and is composed of a unique ID, a target lemma, its part of speech, two sentential contexts in which the target lemma occurs, and positional indices for retrieving the target words in each sentence. In

<sup>1</sup>Each lexeme corresponds to a lemma and its part of speech.

ID	Lemma	POS	Start	End	Sentence
training.en-en.624	leave	VERB	47	51	As mentioned, it was clear that people usually <b>left</b> their homelands in search of a better life.
			13	17	It should be <b>left</b> entirely to the parties to a dispute to choose the modalities of settlement they deemed most appropriate.
training.en-en.625	leave	VERB	47	51	As mentioned, it was clear that people usually <b>left</b> heir homelands in search of a better life.
			80	87	However, no hasty conclusion should be drawn that the Republic of Macedonia was <b>leaving</b> no room for future improvement.

Table 2: Excerpt from the multilingual dataset (En-En): two sentence pairs sharing the same first sentence are shown, with the target word occurrence in bold type.

ID	Tag
training.en-en.624	F
training.en-en.625	F

Table 3: Example of gold file.

both sub-tasks, for each lexeme, we provide two different instances which share one sentence<sup>2</sup>. We provide training and development data only for the multilingual sub-task, whereas test data is provided for both sub-tasks. While training data is produced only in English, both the development and the test data are available in other languages as well. Table 1 provides an overview of the composition of the dataset, which we detail further in the remainder of this paper. Compared to existing datasets, MCL-WiC makes it possible to perform a thorough, high-quality evaluation of a multitude of approaches, ranging from architectures based on pre-trained language models to traditional WSD systems.

In the following, we introduce the multilingual and cross-lingual sub-tasks. Then, we describe the data sources, the selection of the target lexemes and sentence pairs and, finally, the annotation process.

### 3.1 Multilingual sub-task

This sub-task allows systems to be evaluated in a scenario in which only one language at a time is considered. To this end, we manually select sentence pairs in the following language combinations:

<sup>2</sup>To speed up the annotation process, for each lexeme, we selected a fixed sentence and annotated two other sentences so as to obtain two instances.

Ar-Ar, En-En, Fr-Fr, Ru-Ru and Zh-Zh. The multilingual sub-task includes training, development and test splits as reported in Table 1 (top). The training data, available only in English, contains 4000 unique lexemes and 8000 sentence pairs. Instead, both the development and test data splits include 500 unique lexemes and 1000 sentence pairs for each of the aforementioned language combinations. To avoid any bias, each dataset contains a balanced number of tags, i.e. 50% True (T) and 50% False (F).

In Table 2,<sup>3</sup> we report two instances derived from En-En, which share the first sentence. Given the target lemma *leave*, its part of speech (verb) and two sentences in which two occurrences of *leave* are contained, participating systems are required to determine whether the target occurrences (shown in bold type in the Table) share the same meaning (T) or not (F). Since the senses of the target occurrences differ in both sentence pairs, they are both tagged with F in the gold file, as shown in Table 3. Note that, in MCL-WiC, target occurrences can be inflected forms of the target lemma.

### 3.2 Cross-lingual sub-task

The cross-lingual sub-task allows systems to be tested and compared in a cross-lingual scenario. Here, sentence pairs are composed of a sentence in English and a sentence in one of the other MCL-WiC languages, including the following language combinations: En-Ar, En-Fr, En-Ru and En-Zh. It is worth mentioning that, in contrast to past efforts,

<sup>3</sup>Due to space limits we removed some words from the sentences reported in Table 2 and 4.

ID	Lemma	POS	Start	End	Sentence
test.en-ru.18	light	NOUN	46	51	Using a technique for concentrating the solar <b>light</b> , resulted in an overall efficiency of 20%.
			39	50	Каждый представитель может выступать в <b>зависимости</b> от полученных указаний.
test.en-ru.19	light	NOUN	46	51	Using a technique for concentrating the solar <b>light</b> , resulted in an overall efficiency of 20%.
			2	8	С <b>учетом</b> работы, оратор считает целесообразным изложить принципы.

Table 4: Excerpt from the cross-lingual dataset (En-Ru): two sentence pairs sharing the same first sentence are shown, with the target word occurrence in bold type.

all sentences are manually selected and annotated, and that Arabic and Russian are included in a Word-in-Context dataset for the first time.

We report two cross-lingual instances (sentence pairs) in Table 4 for the En-Ru language combination, which share the first sentence. Given the English lemma *light*, its part of speech (noun), and two sentences, one in English where *light* occurs and one in Russian where a translation of *light* appears, participants are asked to determine whether the target occurrence (in bold in the Table) of *light* and its translations into Russian *зависимости* and *учетом* share the same meaning or not. Importantly, translations are allowed to be multi-word expressions and periphrases.

The cross-lingual sub-task comprises test data only and includes 500 unique English lexemes and 1000 sentence pairs for each language combination as reported in Table 1 (bottom). Note that, in this case, all cross-lingual datasets share the same English target lexemes. Similarly to its multilingual counterpart, the data in this sub-task contains a balanced number of T (50%) and F (50%) tags.

### 3.3 Selection of the data and annotation

**Sources of the data** In order to construct MCL-WiC, we leveraged three resources. First, we used the BabelNet<sup>4</sup> multilingual semantic network (Navigli and Ponzetto, 2010) to obtain a set of lexemes in all languages of interest. Subsequently, we extracted sentence pairs containing occurrences of such lexemes from two corpora, namely the United Nations Parallel Corpus (Ziemski et al., 2016, UNPC)<sup>5</sup> and Wikipedia<sup>6</sup>. UNPC is a collection of official records and parliamentary docu-

ments of the United Nations available in the six UN languages<sup>7</sup>, whereas Wikipedia is a wide-coverage multilingual collaborative encyclopedia. These corpora were selected due to their wide coverage in terms of domains and languages. In fact, such heterogeneity allowed for the creation of a new competitive benchmark capable of evaluating the generalization ability of a system in discriminating senses in different domains and across languages. With this aim in view, we derived 50% of the selected sentence pairs from UNPC and the remaining 50% from Wikipedia.

**Selection of lexemes** Starting from BabelNet, we extracted a set of 5250 unique ambiguous lexemes in English and 1000 unique lexemes for each of the following languages: Arabic, Chinese, French and Russian. The selected pairs in English were distributed as follows: 4000 for the training data, 500 for the development data and 750 for the test data (500 for the multilingual sub-task and 250 for the cross-lingual sub-task<sup>8</sup>; we enriched the latter with additional 250 pairs derived from the multilingual test data). Instead, the selected pairs in languages other than English were included in the multilingual sub-task only and distributed as follows: 500 for the development data and 500 for the test data. We selected the target lexemes starting from basic vocabulary words and such that they had at least three senses in BabelNet. A key goal was to cover all open-class parts of speech, namely nouns, verbs, adjectives and adverbs, whose distribution in MCL-WiC is shown in Table 5. The target lexemes were chosen so as to avoid phrasal verbs and multi-word expressions.

<sup>4</sup><https://babelnet.org/>

<sup>5</sup><https://conferences.unite.un.org/uncorpus/>

<sup>6</sup><https://wikipedia.org>

<sup>7</sup>Arabic, Chinese, English, French, Spanish and Russian.

<sup>8</sup>We recall that, in the cross-lingual sub-task, the target lexemes are provided in English and shared across all datasets.

	En-En			Ar-Ar		Fr-Fr		Ru-Ru		Zh-Zh		En-*
	Train	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Test
NOUN	4124	582	528	490	494	548	514	572	582	520	554	458
VERB	2270	246	298	428	398	262	272	352	372	330	364	320
ADJ	1430	158	144	72	98	156	184	54	30	122	62	178
ADV	176	14	30	10	10	34	30	22	16	28	20	44

Table 5: Part-of-speech distribution in MCL-WiC. \* indicates all languages supported in MCL-WiC other than English.

**Selection and annotation of sentence pairs** For each of the target lexemes, we annotated two sentence pairs from either UNPC or Wikipedia. All selected sentences were well-formatted and, most importantly, provided a sufficient semantic context to determine the meaning of the target occurrences unequivocally. Subsequently, each sentence pair was associated with a tag, depending on whether the target words in the two contexts are used with the same meaning (T) or not (F). To perform both the selection of the data as well as the annotation, we employed eight annotators with a high level of education and linguistic proficiency in the corresponding language; the annotation work required approximately six months. Importantly, all annotators followed specific criteria which we describe in the following paragraph.

**Annotation criteria** We provided each annotator with general annotation guidelines. Besides general criteria, each annotation team<sup>9</sup> established ad-hoc guidelines for specific linguistic issues, some of which will be briefly illustrated in Section 4, below.

General annotation criteria can be broadly divided into grammatical and lexicographic-semantic criteria. The former refer to the format and the grammatical correctness of the sentences to be selected: annotators were asked to choose well-written sentences only, i.e. sentences with a clear structure, ending with a full stop and containing a main clause. Instead, lexicographic-semantic criteria refer to the attribution of the labels. To determine whether two occurrences were used with the same meaning or not, annotators were asked to use multiple reputable dictionaries (e.g. for English we used the Merriam-Webster, Oxford Dictionary of English and English Collins dictionaries). Moreover, to avoid misperceptions in the same-sense tagging annotations, we asked annotators to justify

<sup>9</sup>An annotation team is made up of annotators working on the same language.

their choices by providing substitutes for the target occurrences with synonyms, hypernyms, paraphrases or the like. Contrary to what was done in WiC and XL-WiC, we argue that, for the purposes of this task, annotating according to lexicographic motivations, i.e. by using reliable dictionaries, contributes significantly to minimizing the impact of subjectivity, thus producing more adequate and consistent data. Finally, lexicographic-semantic criteria also provided concrete indications and examples regarding the attribution of tags. For instance, T was used if and only if the two target occurrences were used with exactly the same meaning or, in other words, if, using a dictionary, the definition of the two target words was the same.

**Inter-annotator agreement** In order to determine the degree of uncertainty encountered during the annotation process, we computed the inter-annotator agreement. To this end, we randomly selected a sample of 500 sentence pairs from each of the En-En and Ru-Ru multilingual datasets, and 200 sentence pairs from the En-Ar and En-Zh cross-lingual datasets. Validators were provided with the same guidelines used during the annotation process. We calculated the agreement between two different annotators using the Cohen’s kappa, obtaining  $\kappa=0.968$  in En-En, 0.952 in Ru-Ru, 0.94 in En-Ar and 0.91 in En-Zh, which is interpreted as almost perfect agreement.

**Data format** For each sub-task, we provide two types of file (.data and .gold) in JSON format. The .data files contain the following information: a unique ID, the lemma, its part of speech, the two sentences and the positional indices to identify the target occurrences to be considered (see Tables 2 and 4). Instead, the .gold files include the gold answers, i.e. the corresponding ID and tag, as shown in Table 3.



## 4 Linguistic Issues

In this section, we describe interesting language-specific issues which required additional guidelines. Due to space limits, we focus on languages which do not use the Latin alphabet, i.e. Arabic, Chinese and Russian, illustrating only the most significant issues encountered.

**Arabic** From a WSD perspective, compared to other languages, written Arabic poses bigger challenges due to the omission of vocalization, which increases the degree of semantic ambiguity. In fact, the vocalization, expressed by diacritics placed above or below consonants, contributes significantly to determining the right interpretation and thus the meaning of words. For instance, the unvocalized word form *b-r-d* could be interpreted as *bard* (“cold”), *burd* (“garment”) or *barad* (“hail”). Of course, in Arabic, polysemy also affects vocalized words, which can have multiple meanings, e.g. *ummiyy* means “maternal”, but also “illiterate”. For the purposes of MCL-WiC, we chose to keep the sentences as they are found in UNPC and Wikipedia, i.e. unvocalized in the vast majority of cases, while – instead – providing the target lemmas in the vocalized form. This was done in order to avoid lexical ambiguity deriving from lemmas which share the same word form but are vocalized in a different way. Furthermore, this choice facilitated the selection and annotation of sentence pairs in which a given target lemma occurs.

**Chinese** Since Chinese does not adopt an alphabet, the semantic ambiguity that can be found in English homographs is basically lost. In Chinese, if two unrelated words are pronounced in the same way, such as “plane” (the airplane) and “plane” (the surface), they are not usually written in the same way. By way of illustration, 沉默, meaning “silent; to be silent” and 沉没, “to sink”, are both pronounced as *chénmò*, but, because they are written with different characters, they cannot be considered ambiguous words. Analogously, some characters have an extremely high semantic ambiguity themselves, but since they appear most frequently in polysyllabic words, their ambiguity is lost. For example, the character *guǒ* 果 has at least two meanings, “fruit” and “result”, but this character almost never stands as a word on its own in contemporary Chinese. In the current lexicon most of the Chinese words are composed of two or more characters; when it appears in actual texts, *guǒ* is al-

most always connected to other characters, and the word thus formed is no longer semantically ambiguous. Finally, similarly to the cross-lingual sub-task, some ambiguity had to be discarded in translation, as in the case of Chinese classifiers which have a marked potential for semantic ambiguity. For example, *dào* 道 is, among others, the classifier for long and narrow objects, as in *yī dào hé* 一道河, a river (one+classifier+river), or for doors, walls and similar objects with an entry and an exit, as in *yī dào mén* 一道门, a door (one+classifier+door). However, since classifiers are virtually absent in European languages, they could not be applied in the cross-lingual sub-task and were discarded.

**Russian** A noteworthy issue encountered by Russian annotators concerned the verbal aspects which can be viewed as one of the most challenging features of the Russian language especially for L2-learners<sup>10</sup> with no Slavic background. In Russian, a verb can be perfective, imperfective or both. Normally, a perfective verb has one or more imperfective counterparts and vice versa. Broadly speaking, perfective verbs are typically used to express non-repetitive actions completed in the past, or actions which will certainly be carried out in the future, and also in general for past or future actions for which the speaker intends to emphasize the result that was or will be achieved. Conversely, imperfective verbs are used to express actions which are incomplete, habitual, in progress, or actions for which the speaker does not stress the result to be attained. In MCL-WiC, given a verbal target lexeme, we decided to choose sentences in which the target words occurring in the selected sentences and the target lemma shared the same aspect. In fact, in Russian, although pairs of perfective and imperfective verbs such as *делать, сделать* (to do) or *спрашивать, спросить* (to ask) show a high degree of morphological relatedness, they tend to be considered as distinct lemmas.

Another interesting issue regards participles. In some cases, annotators raised issues concerning the part of speech of participles occurring as target words in the selected sentences. In fact, Russian participles derive from verbs, but are declined and can behave as adjectives. Since the target lexemes and the corresponding occurrences must share the same part of speech, we decided to discard sentences in which the part of speech of the target

<sup>10</sup>In language teaching, L2 indicates a language which is not the native language of the speaker.

words could not be determined unequivocally.

## 5 Participating Systems

This Section is devoted to the participating systems. First, we briefly describe the rules of the competition. Subsequently, we provide an overview of the data and approaches used by participants. Then, we focus on some of the best-scoring systems and provide a breakdown of the techniques adopted. We report the three best-performing teams for each sub-task and language combination in Tables 6 and 7. All results are publicly available on the official MCL-WiC page on GitHub<sup>11</sup>. For each winning team, we show only the best performance in the corresponding category.

### 5.1 Rules of the competition

Participants were given no constraints as far as data was concerned; for instance, the development data could be used for training or it was allowed to enrich the provided data by constructing new datasets in an automatic or semi-automatic fashion. Furthermore, we allowed more than one participant for each team. Participating teams could upload up to five submissions, each including up to 9 language combinations for the two sub-tasks.

### 5.2 Data

**Multilingual sub-task** As far as English is concerned, the majority of participating systems used the MCL-WiC training and development data. Some participants also used the data derived from WiC and XL-WiC. Furthermore, automatically-constructed WiC-like datasets were obtained by some participants, starting from semantic resources such as SemCor (Miller et al., 1993), WordNet and the Princeton WordNet Gloss Corpus (PWNG)<sup>12</sup>, or by automatically translating available datasets into English. The available data was also enriched via sentence reversal augmentation (given a sentence pair, the two sentences were swapped). In some cases, the development and trial<sup>13</sup> data was used to enrich the training data.

As regards languages other than English, most participants used XL-WiC data, or new training and development datasets were obtained by splitting the MCL-WiC language-specific development

data. Alternatively, in zero-shot scenarios, participants trained their models using the English training data. Furthermore, some participants augmented the training and development data by including the trial data. Also in this case, training and development splits were augmented via sentence reversal.

**Cross-lingual sub-task** In the cross-lingual sub-task, most participants used the MCL-WiC English training and development data in zero-shot settings. A smaller group of participants used WiC and XL-WiC data. Some participants created additional training and development data from other resources such as the Open Multilingual WordNet and PWNG. Additional training and development data was produced via Machine Translation.

### 5.3 Approaches

**Multilingual sub-task** Most participants used XLM-RoBERTa (Conneau et al., 2020) as pre-trained language model to obtain contextual representations of the target occurrences. Other models frequently used by participants were mBERT, RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2019), ELECTRA (Clark et al., 2019) and ERNIE (Sun et al., 2020). The majority of participants made use of fine-tuned contextualized embeddings and used logistic regression to perform binary classification. Some participants used ensembles and majority voting.

**Cross-lingual sub-task** Also in this sub-task, XLM-RoBERTa was the most used multilingual language model. Again, the majority of systems obtained contextualized embeddings, passing them to a logistic regression unit. In this case, participants mainly explored zero-shot approaches. Some participants made use of ensembles, adversarial training, pseudo-labelling (Wu and Prasad, 2017) and cross-validation techniques.

### 5.4 Competition and best-scoring systems

The MCL-WiC competition took place on the CodaLab<sup>14</sup> open Web-based platform and reported 170 participants, out of which 48 uploaded one or more datasets. Overall, 170 submissions were received, the majority of which were focused on the multilingual sub-task and specifically on the En-En dataset. As far as the evaluation metric was concerned, systems were tested using the accuracy

<sup>11</sup><https://github.com/SapienzaNLP/mcl-wic>

<sup>12</sup><http://wordnetcode.princeton.edu/>

<sup>13</sup>As trial data, we provided 4 instances for each sub-task and dataset.

<sup>14</sup><https://competitions.codalab.org/competitions/27054>

Dataset	Team	Score
<b>Ar-Ar</b>	Cam	<b>84.8</b>
	LIORI	84.6
	MCL@IITK; DeathwingS	84.5
<b>En-En</b>	MCL@IITK; oyx	<b>93.3</b>
	zhestyatsky	92.7
	Cam	92.5
<b>Fr-Fr</b>	MCL@IITK	<b>87.5</b>
	Cam	86.5
	LIORI	86.4
<b>Ru-Ru</b>	Cam	<b>87.4</b>
	LIORI	86.6
	godzilla	86.5
<b>Zh-Zh</b>	stce	<b>91.0</b>
	godzilla	90.8
	PALI	90.5

Table 6: Multilingual section: five best-scoring systems by language combination.

score. In what follows, we provide insights regarding the approaches adopted by some of the best-performing participating systems, based on the information we received.

**Cam** The Cam team (Yuan and Strohmaier, 2021) made use of the WiC and XL-WiC datasets in addition to the MCL-WiC data. Furthermore, examples from the Sense Complexity Dataset (Strohmaier et al., 2020, SeCoDa) and the Cambridge Advanced Learner’s Dictionary (CALD) were extracted. Cam used pre-trained XLM-RoBERTa as underlying language model and added two additional layers on top to perform binary classification with tanh and sigmoid activation, respectively. As input, the following items were concatenated: the representation corresponding to the first token of the sequence, the representations of the target words in both sentences, as well as the absolute difference, cosine similarity and pairwise distance between the two vectors. When the target word was split into multiple sub-tokens, Cam took the average representation rather than the first sub-token. Finally, a two-step training strategy was applied: 1) pre-training the system using out-of-domain data, i.e. WiC, XL-WiC, SeCoDa and CALD; 2) fine-tuning the system on MCL-WiC data.

**godzilla** godzilla enriched the MCL-WiC training data by automatically constructing a dataset starting from WordNet and using Machine Translation. Different types of pre-trained models, such

as RoBERTa and XLM-RoBERTa, were adopted. godzilla highlighted the target words by surrounding them with special markings on both sides and appending the target words to the end of each sentence. As architecture, this system used the next sentence prediction models from the hugging face<sup>15</sup> library. Given the strong connection between En-Ar, En-Fr, En-Ru, En-Zh test datasets, pseudo-tagging was used for each language combination. Finally, godzilla applied label smoothing and model merging.

**LIORI** The LIORI<sup>16</sup> team (Davletov et al., 2021) used the datasets provided in the MCL-WiC competition. Specifically, the training data was enriched with 70% of the development data for Arabic, Chinese, French and Russian, and the whole trial data. Optionally, data augmentation was performed by swapping sentences in each example. LIORI fine-tuned XLM-RoBERTa on a binary classification task and used a 2-layered feed-forward neural network on top of the language model with dropout and the tanh activation function. Sentences in each pair were concatenated by the special token "</s>" and fed to XLM-RoBERTa. As input, the model took the concatenation of the contextualized embeddings of the target words, aggregating over sub-tokens either by max pooling, or just by taking the first sub-token. LIORI used a voting ensemble composed of three models: the first model trained with data augmentation, using the concatenations of the first sub-tokens of the target words; the second trained with data augmentation using max-pooling over sub-tokens; finally, the third trained without data augmentation and using concatenations of the first sub-tokens.

**stce** stce used the MCL-WiC datasets and built additional training data using HowNet (Dong and Dong, 2003). Furthermore, the training data was enriched by pseudo-labelling the test datasets. Data cleaning was performed and target words were surrounded by special markings. The main language model used was XLM-RoBERTa-large. During the training process, dynamic negative sampling was performed for each batch of data fed to the model. At the same time, stce adopted the Fast Gradient Method and added disturbance to the embedding layer to obtain more stable word representations.

<sup>15</sup><https://huggingface.co/>

<sup>16</sup>The following member of the team LIORI took part in the competition: davletov.

Dataset	Team	Score
<b>En-Ar</b>	PALI	<b>89.1</b>
	godzilla	87.0
	Cam; LIORI	86.5
<b>En-Fr</b>	PALI	<b>89.1</b>
	godzilla	87.6
	LIORI	87.2
<b>En-Ru</b>	PALI	<b>89.4</b>
	godzilla	88.5
	RyanStark; rxy1212	87.3
<b>En-Zh</b>	PALI; RyanStark	<b>91.2</b>
	Cam	88.8
	MagicPai	88.6

Table 7: Cross-lingual sub-task: three best-scoring systems by language combination.

**zhestyatsky** Zhestiankin and Ponomareva (2021) augmented the English MCL-WiC training and development data with WiC. Training and development data were split randomly to create a larger training sample which included 97.5% of the data, while leaving only 2.5% for the new development dataset. Then, bert-large-cased embeddings were fine-tuned using AdamW as optimizer with a learning rate equal to  $1e-5$ . Each sentence was split by BertTokenizerFast into 118 tokens maximum. The model was trained for 4.5 epochs and stopped by Early Stopping with patience equal to 2. For each sentence, zhestyatsky took the embeddings of all sub-tokens corresponding to the target word and max pooled them into one embedding. Subsequently, zhestyatsky evaluated the cosine similarity of these embeddings and activated this value through ReLU.

**MCL@IITK** First, the MCL@IITK<sup>17</sup> team (Gupta et al., 2021) pre-processed the sentences by adding a signal, either double quotes on both sides of the target word, or the target word itself appended to the end of the sentence. For En-En, MCL@IITK enriched the MCL-WiC training data using sentence reversal augmentation, WiC and SemCor. MCL@IITK obtained embeddings of the target words using the last hidden layer, and passed them to a logistic regression unit. MCL@IITK used ELECTRA, ALBERT, and XLM-RoBERTa as language models and submitted probability sum ensembles. For the non-English multilingual sub-task, MCL@IITK used XLM-RoBERTa only and

<sup>17</sup>The following members of the MCL@IITK team took part in the competition: jaymundra, rohangpt and dipakam.

tackled all four language pairs jointly. A 9:1 train-dev split with sentence reversal augmentation was used on the non-English dev data, in addition to En-En train data and XL-WiC with an ensemble model. For the cross-lingual subtask, ELECTRA embeddings were used. The models were trained on partly back-translated En-En train set and validated on back-translated En-En development set.

**PALI** The PALI<sup>18</sup> team (Xie et al., 2021) enriched the MCL-WiC data using WordNet while keeping the original cross-lingual data to maintain the target words in the cross-lingual data. After text pre-processing, task-adaptive pre-training was performed using the MCL-WiC data. The target words were surrounded by special symbols. PALI used XLM-RoBERTa as main language model and took its final output layer, concatenating the [CLS] token with the embeddings of the target occurrences in each sentence pair. To increase the training data, PALI exchanged the order of 20% of the sentence pairs. During training, lookahead (AdamW) was used together with adversarial training implemented by the Fast Gradient Method to obtain more stable word representations. Hyperparameters were tuned through trial-and-errors. The models of stratified 5-fold cross-validation were averaged to yield the final prediction results.

## 6 Baselines

Following Raganato et al. (2020), we used a baseline transformer-based binary classifier. Thus, first, given a sentence pair, a dense representation is obtained for each target occurrence. As indicated in Devlin et al. (2019), in the case that a target occurrence is split into multiple sub-tokens, the first sub-token is selected. The resulting representations are then given as input to a binary classifier implemented following Wang et al. (2019). We selected the Adam optimizer (Kingma and Ba, 2015) with learning rate and weight decay equal to  $1e-5$  and 0, respectively, and trained for 10 epochs.

We experimented with two different contextualized embedding models: BERT (base-multilingual-cased) and XLM-RoBERTa (base). As for the data, in contrast to most participants, we made use of the data provided for the task only. We used En-En as training and development data for English. As for other language combinations, we trained on En-En and validated both on En-En or and on the other

<sup>18</sup>The following members of the PALI team took part in the competition: endworld and xsysigma.

Model	Ar-Ar	En-En	Fr-Fr	Ru-Ru	Zh-Zh	En-Ar	En-Fr	En-Ru	En-Zh
mBERT <sub>1</sub>	<b>76.2</b>	84.0	<b>78.7</b>	74.5	77.5	65.9	71.6	68.2	<b>68.9</b>
XLMR-base <sub>1</sub>	75.4	<b>86.6</b>	77.9	<b>76.5</b>	<b>78.5</b>	<b>67.7</b>	<b>71.8</b>	<b>74.2</b>	66.1
mBERT <sub>2</sub>	<b>76.4</b>	84.0	<b>78.7</b>	74.6	76.6	62.0	69.4	66.7	64.2
XLMR-base <sub>2</sub>	75.4	<b>86.6</b>	77.7	<b>76.5</b>	<b>78.9</b>	<b>67.7</b>	<b>74.9</b>	<b>74.2</b>	<b>71.3</b>

Table 8: Accuracy of baselines for multilingual and cross-lingual sub-tasks. Columns indicate the test set used. In setting 1, we used the En-En training data and the En-En development data. In setting 2, we used the En-En training data and the corresponding development datasets in languages other than English.

language multilingual development data. Table 8 reports the best training results according to the corresponding validation.

## 7 Results and Discussion

In this section, we discuss the results achieved in our competition. Overall, the MCL-WiC dataset allows systems to attain high performances, in the 85-93% accuracy range. This leads us to hypothesize that, in general, systems were able to develop a good ability in capturing sense distinctions without relying on a fixed sense inventory.

When compared to the proposed baselines, we observe that best-performing systems were able to achieve an absolute improvement of up to 27.1 points over the corresponding baselines (e.g. on En-Ar, cf. Tables 7 and 8). Both our baselines and the systems developed by participants confirm that, in this task, XLM-RoBERTa outperforms BERT in most language combinations. The highest score was obtained in En-En, with the best system achieving 93.3% accuracy. Note that our baselines were also able to attain good performances in En-En, i.e. 84.0% using BERT and 86.6% with XLM-RoBERTa, without benefiting from additional training and development data. Interestingly, Chinese was the language which achieved the second-best results, both in Zh-Zh and En-Zh, attaining on average results which were considerably higher. Instead, Arabic seems to have been the most difficult language for participants, especially in Ar-Ar. A reason for this result, deserving further exploration, could lie in morpho-semantic features inherent in Arabic, which we briefly outlined in Section 4.

Zero-shot approaches differ in the performances achieved by participants in the two sub-tasks: in the cross-lingual sub-task participants were able to achieve slightly better performances than those in the multilingual setting, most probably thanks to the presence of English in both the training and

the test data, and, more in general, to the availability of English WiC-style datasets which could be used to enrich the already provided data. With the exception of Chinese, instead, on the multilingual sub-task we observe a performance drop between 1.6 and 4.3%.

Finally, we note that performance boosts were observed across the board when using data augmentation, especially by swapping the two sentences within a pair or by coupling the second sentences of two pairs sharing the same first sentence and the same meaning. Another consistent performance increase, observed both in the multilingual and in the cross-lingual sub-task, was obtained when adding a signal on both sides of the target occurrences.

## 8 Conclusions

In this paper, we described the SemEval-2021 Task 2 and introduced Multilingual and Cross-lingual Word-in-Context (MCL-WiC), the first entirely manually-curated WiC-style dataset in five European and non-European languages, namely Arabic, Chinese, English, French and Russian. MCL-WiC allows the inherent ability of systems to discriminate between word senses within the same language to be tested, and also, interestingly, within cross-lingual scenarios in which a system is evaluated in two languages at the same time, namely English and one of the remaining MCL-WiC languages.

While current Word-in-Context datasets focus primarily on single tokens, as a suggestion for future work we would like to further explore the integration of multi-word expressions and idiomatic phrases into a Word-in-Context task. This would allow us to investigate the intrinsic ability of a system to correctly discriminate the semantics of such linguistic constructs, especially those whose meaning is not compositional, i.e. it cannot be derived by combining the meaning of each of their individual components.

## Acknowledgments

The authors gratefully acknowledge the support of the ERC Consolidator Grant MOUSSE No. 726487 and the



ELEXIS project No. 731015 under the European Union’s Horizon 2020 research and innovation programme.



We gratefully thank Luisa Borchio, Ibraam Abdelsayed, Anna Guseva, Zhihao Lyu and Beatrice Buselli for their valuable annotation work.

## References

- Carlos Santos Armendariz, Matthew Purver, Senja Pollak, Nikola Ljubešić, Matej Ulčar, Ivan Vulić, and Mohammad Taher Pilehvar. 2020. [SemEval-2020 Task 3: Graded Word Similarity in Context](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 36–49.
- Carlos Santos Armendariz, Matthew Purver, Matej Ulčar, Senja Pollak, Nikola Ljubesic, Marko Robnik-Sikonja, Mark Granroth-Wilding, and Kristiina Vaik. [CoSimLex: A resource for evaluating graded word similarity in context](#). In *Proceedings of The 12th Language Resources and Evaluation Conference*, page 5878–5886.
- Agostina Calabrese, Michele Bevilacqua, and Roberto Navigli. 2020. [EViLBERT: Learning task-agnostic multimodal sense embeddings](#). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 481–487. International Joint Conferences on Artificial Intelligence Organization.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. [Electra: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Édouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451.
- Adis Davletov, Nikolay Arefyev, Denis Gordeev, and Alexey Rey. 2021. [LIORI at SemEval-2021 Task 2: Span Prediction and Binary Classification approaches to Word-in-Context Disambiguation](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Zhendong Dong and Qiang Dong. 2003. [HowNet-a hybrid language and knowledge resource](#). In *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, pages 820–824. IEEE.
- Rohan Gupta, Jay Mundra, Deepak Mahajan, and Ashutosh Modi. 2021. [MCL@IITK at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation using Augmented Data, Signals, and Transformers](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok, Thailand. Association for Computational Linguistics.
- Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. [Improving word representations via global context and multiple word prototypes](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. [Sensembed: Learning sense embeddings for word and relational similarity](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 95–105.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A Method for Stochastic Optimization](#). In *International Conference on Learning Representations*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [ALBERT: A Lite BERT for Self-supervised Learning of Language Representations](#). In *International Conference on Learning Representations*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Daniel Loureiro and Alípio Jorge. 2019. [Language modelling makes sense: Propagating representations through wordnet for full-coverage word sense disambiguation](#). In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5682–5691. Association for Computational Linguistics.

- Diana McCarthy and Roberto Navigli. 2007. **SemEval-2007 Task 10: English lexical substitution task**. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, page 48–53.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. **Context2Vec: Learning generic context embedding with bidirectional LSTM**. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 51–61. ACL.
- Rada Mihalcea, Ravi Sinha, and Diana McCarthy. 2010. **SemEval-2010 Task 2: Cross-lingual lexical substitution**. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 9–14.
- George A. Miller, R.T. Beckwith, Christiane D. Fellbaum, D. Gross, and K. Miller. 1990. **Introduction to WordNet: an online lexical database**. *International Journal of Lexicography*, 3(4).
- George A. Miller, Claudia Leacock, Randee Teng, and Ross T. Bunker. 1993. **A semantic concordance**. In *Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.
- Roberto Navigli. 2009. **Word sense disambiguation: A survey**. *ACM computing surveys (CSUR)*, 41(2):1–69.
- Roberto Navigli and Simone Paolo Ponzetto. 2010. **BabelNet: Building a very large multilingual semantic network**. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.
- Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. **Efficient non-parametric estimation of multiple embeddings per word in vector space**. *CoRR*, abs/1504.06654:1059–1069.
- Maria Pelevina, Nikolay Arefiev, Chris Biemann, and Alexander Panchenko. 2016. **Making sense of word embeddings**. In *Proceedings of the 1st Workshop on Representation Learning for NLP, Rep4NLP@ACL 2016, Berlin, Germany, August 11, 2016*, pages 174–183. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and José Camacho-Collados. 2019. **WiC: the Word-in-Context dataset for evaluating context-sensitive meaning representations**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1267–1273. Association for Computational Linguistics.
- Alessandro Raganato, Tommaso Pasini, Jose Camacho-Collados, and Mohammad Taher Pilehvar. 2020. **XI-wic: A multilingual benchmark for evaluating semantic contextualization**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7193–7206.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020a. **SensEmBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation**. In *Proc. of AAAI*, pages 8758–8765.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020b. **With More Contexts Comes Better Performance: Contextualized Sense Embeddings for All-Round Word Sense Disambiguation**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, page 3528–3539. Association for Computational Linguistics.
- David Strohmaier, Sian Gooding, Shiva Taslimipoor, and Ekaterina Kochmar. 2020. **SeCoDa: Sense complexity dataset**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5962–5967, Marseille, France. European Language Resources Association.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. **Ernie 2.0: A continual pre-training framework for language understanding**. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. **SuperGLUE: A stickier benchmark for general-purpose language understanding systems**. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 3261–3275.
- Hao Wu and Saurabh Prasad. 2017. **Semi-supervised deep learning using pseudo labels for hyperspectral image classification**. *IEEE Transactions on Image Processing*, 27(3):1259–1270.
- Shuyi Xie, Jian Ma, Haiqin Yang, Lianxin Jiang, Yang Mo, and Jianping Shen. 2021. **PALI at SemEval-2021 task 2: Fine-Tune XLM-RoBERTa for Word in Context Disambiguation**. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Zheng Yuan and David Strohmaier. 2021. **Cambridge at SemEval-2021 Task 2: Neural WiC-Model with Data Augmentation and Exploration of Representation**. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Boris Zhestiankin and Maria Ponomareva. 2021. **Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning**. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.

Michał Ziemiński, Marcin Junczys-Dowmunt, and Bruno Pouliquen. 2016. [The United Nations Parallel Corpus v1.0](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3530–3534.



# SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning

Boyuan Zheng<sup>2\*</sup>, Xiaoyu Yang<sup>1</sup>, Yu-Ping Ruan<sup>3</sup>, Zhenhua Ling<sup>3</sup>,  
Quan Liu<sup>3</sup>, Si Wei<sup>4</sup>, Xiaodan Zhu<sup>1</sup>

<sup>1</sup>Queen’s University, Kingston, Canada; <sup>2</sup>Northeastern University, Shenyang, China

<sup>3</sup>University of Science and Technology of China; <sup>4</sup>iFlytek Research, Hefei, China

steven.zheng010@gmail.com; xiaoyu.yang@queensu.ca;

{quanliu, zhling}@ustc.edu.cn; siwei@iFlytek.com.cn;

xiaodan.zhu@queensu.ca

## Abstract

This paper introduces the SemEval-2021 shared task 4: Reading Comprehension of Abstract Meaning (ReCAM). This shared task is designed to help evaluate the ability of machines in representing and understanding abstract concepts. Given a passage and the corresponding question, a participating system is expected to choose the correct answer from five candidates of abstract concepts in a cloze-style machine reading comprehension setup. Based on two typical definitions of *abstractness*, i.e., the *imperceptibility* and *nonspecificity*, our task provides three subtasks to evaluate the participating models. Specifically, Subtask 1 aims to evaluate how well a system can model concepts that cannot be directly perceived in the physical world. Subtask 2 focuses on models’ ability in comprehending *nonspecific* concepts located high in a hypernym hierarchy given the context of a passage. Subtask 3 aims to provide some insights into models’ generalizability over the two types of abstractness. During the SemEval-2021 official evaluation period, we received 23 submissions to Subtask 1 and 28 to Subtask 2. The participating teams additionally made 29 submissions to Subtask 3. The leaderboard and competition website can be found at <https://competitions.codalab.org/competitions/26153>. The data and baseline code are available at <https://github.com/boyuanzheng010/SemEval2021-Reading-Comprehension-of-Abstract-Meaning>.

## 1 Introduction

Humans use words with abstract meaning in their daily life. In the past, research efforts have been exerted to better understand and model abstract meaning (Turney et al., 2011; Theijssen et al.,

\* This work was performed when Boyuan Zheng visited Queen’s University.

2011; Changizi, 2008; Spreen and Schulz, 1966). Modelling abstract meaning is closely related to many other NLP tasks such as reading comprehension, metaphor modelling, sentiment analysis, summarization, and word sense disambiguation.

In the past decade, significant advancement has been seen in developing computational models for semantics, based on deep neural networks. In this shared task, we aim to help assess the capability of the state-of-the-art deep learning models on representing and modelling abstract concepts in a specific reading comprehension setup.

We introduce SemEval-2021 Task 4, Reading Comprehension of Abstract Meaning (**ReCAM**). Specifically, we design this shared task by following the machine reading comprehension framework (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2016), in which computers are given a passage  $D_i$  as well as a human summary  $S_i$  to comprehend. If a model can digest the passage as humans do, we expect it to predict the abstract word used in the summary, if the abstract word is masked. Unlike the previous work that requires computers to predict concrete concepts, e.g., named entities, in our task we ask models to fill in abstract words removed from human summaries. During the SemEval-2021 official evaluation period, we received 23 submissions to Subtask 1 and 28 submissions to Subtask 2. The participating teams additionally made 29 submissions to Subtask 3. In this paper, we induce the shared task and provide a summary for the evaluation.

## 2 Task Description

We organize our shared task based on two typical definitions of *abstractness*, named as *imperceptibility* and *nonspecificity* in this paper, implemented in Subtask 1 and Subtask 2, respectively. Subtask 3 further evaluates models’ generalizability over the two definitions of abstractness.

<b>Passage</b>	... Observers have even named it after him, “Abenomics”. It is based on three key pillars of monetary policy to ensure long-term sustainable growth in the world’s third-largest economy, with fiscal stimulus and structural reforms. In this weekend’s upper house elections, ...
<b>Question</b>	Abenomics: The <i>@placeholder</i> and the risk.
<b>Answer</b>	(A) chance (B) prospective (C) government (D) <b>objective</b> (E) threat

Table 1: An example for Subtask 1. The correct answer to the question is *objective*.

## 2.1 Subtask 1: ReCAM-Imperceptibility

In one definition (Turney et al., 2011; Theijssen et al., 2011; Spreen and Schulz, 1966), concrete words refer to things, events, and properties that humans can directly perceive with their senses, e.g., *trees* and *flowers*. In contrast, abstract words refer to “ideas and concepts that are distant from immediate perception”, e.g., *objective*, *culture*, and *economy*. In Subtask 1, we perform reading comprehension on *imperceptible* abstract concepts, named as **ReCAM-ImPerceptibility**. Table 1 shows an example.

## 2.2 Subtask 2: ReCAM-NonSpecificity

The second typical definition of abstractness is based on *nonspecific* concepts (Theijssen et al., 2011; Spreen and Schulz, 1966). Compared to specific concepts such as *groundhog* and *whale*, words such as *vertebrate* are regarded as more *abstract*. Our Subtask 2, named as **ReCAM-NonSpecificity**, is designed based on this viewpoint. We will discuss how the datasets are constructed in Section 3.

## 2.3 Subtask 3: ReCAM-Cross

In this subtask, participants are asked to submit their predictions on the test data of Subtask 2, using models trained on the training data of Subtask 1, and vice versa. This subtask aims to demonstrate models’ generalizability between modelling the two typical definitions of abstractness.

## 3 Data Construction

We develop our multi-choice machine reading comprehension datasets based on the XSum summarization dataset (Narayan et al., 2018). We first locate words with abstract meaning using our abstractness scorers. Then we perform data filtering to select our target words to construct our datasets.

## 3.1 The XSum Data

By collecting online articles from the British Broadcasting Corporation (BBC), Narayan et al. (2018) developed a large-scale text summarization dataset, XSum, in which each article has a single sentence summary. We developed our ReCAM dataset based on XSum.

## 3.2 Finding Imperceptible Concepts

**Abstractness Scorer for Imperceptibility** Following Turney et al. (2011), we use the MRC Psycholinguistic Database (Coltheart, 1981), which includes 4,295 words rated with a degree of abstractness by human subjects, to train our abstractness scorer for *imperceptibility*. The rating of the words in the MRC Psycholinguistic Database ranges from 158 (highly abstract) to 670 (highly concrete). We linearly scale the rating to the range of 0 (highly abstract) to 1 (highly concrete). The neural regression model accepts fixed Glove embedding (Pennington et al., 2014) as input and predicts the abstractness rating score between 0 and 1. Our regression model is a three-layer network that consists of two non-linear hidden layers with the ReLU activation and a sigmoid output layer. The mean square error (MSE) is used as the training loss.

To test the regression model’s performance, we randomly split the MRC Psycholinguistic Database into train and test set with the size of 2,148 and 1,877, respectively. Table 2 shows the final performance of the neural regression model on the MRC database. We use the Pearson correlation between ratings predicted by models and original ratings from MRC as the evaluation metric. We can see that the regression model achieves high correlation coefficients (the higher, the better), i.e., 0.934 and 0.835, on the training and test set. The correlations are significant ( $p$ -values are smaller than  $10^{-5}$ ), reflecting the quality of our models in finding abstract words. Note that Turney et al. (2011) report a correlation score of 0.81 on their MRC test set. Their training-test split is unavailable, so we run cross-validation here in our experiment. The scorer can then be used to assign an *imperceptibility* score to a word that is not in the MRC Psycholinguistic Database.

Using the abstractness scorer described above, we assign an abstractness value to each word in summaries and select words with a value lower than 0.35 as the candidates for our *target words* (words that will be removed from the summaries

	#samples	Pearson $r$	$p$ -value
train	2,148	0.934	$p < 10^{-5}$
test	1,877	0.854	$p < 10^{-5}$

Table 2: Fitting performance of neural regression model on the MRC database.

to construct questions). We only consider content words as potential target words, i.e., nouns, verbs, adjectives, and adverbs. For this purpose, we use part-of-speech tagging model (?) implemented in Stanza (Qi et al., 2020).

### 3.3 Finding *Nonspecific* Concepts

**Nonspecificity Scorer** Following the work of Changizi (2008), we assign a *nonspecificity* score to a word token based on the hypernym hierarchy of WordNet (Miller, 1998). Specifically, the root of the hierarchy is at level 0 and regarded as the most abstract. The abstractness of a node in the hierarchy is measured by the maximal length of its path to the root. The hypernym level in WordNet is between 0 and 17. For each word token in summaries, we use Adapted Lesk Algorithm (Banerjee and Pedersen, 2002) to label the sense since the WordNet hypernym hierarchy works at the sense level. Since a summary sentence may be short, we concatenate each summary sentence with the corresponding passage for word sense disambiguation. Built on this, each token, which is labelled with a sense, receives an abstractness score based on the WordNet hierarchy.

Using the *nonspecificity* scorer, we assign an *nonspecificity* value to each word in summaries and select words with a value smaller than six as the candidate target words. The targets words will be nouns and verbs since the hypernym hierarchy in WordNet (?) consists of these two POS types.

### 3.4 Filtering

We aim to avoid developing simple questions. For example, if a target word also appears in the passage, it is likely that a model can easily find the answer without the need to understand the passage in depth.

**Filtering by Lemmas** We lemmatized passages and summaries. If a lemma appears both in a summary and the corresponding passage, the lexemes of the lemma will not be considered as target words. Note that a strict filter may exclude some good candidates for target words but helps avoid introducing

many simple questions.

**Filtering by Synonyms and Antonyms** For a word in a summary, if a synonym or antonym of the word appears in the corresponding passage, we will not consider this word to be our target word. We use WordNet (?) to derive synonyms and antonyms. Instead of using word sense disambiguation (WSD), for a word  $w_i$  in a summary, we use all senses of this word and add all synonyms and antonyms into a pool. Only if none of the words in the pool appear in the passage, we consider  $w_i$  as a candidate target word. Otherwise, we will not use  $w_i$  to construct a question for this passage-summary pair.

**Filtering by Similarity** We further filter words by similarity. For each candidate target word in a summary and each word in the passage, we calculate similarity and use that to perform further filtering.

We use 300-dimension GloVe word embedding trained on 840 billion tokens (Pennington et al., 2014). We calculate the cosine similarity between a candidate target word and a passage word. For contextual embedding, we embed each sentence in a passage as well as the summary into a context-aware representation matrix using the BERT-large uncased language model. Then, we calculate the similarity between each passage token and question token with the cosine similarity. If the similarity is higher than 0.85, we will not consider the involved summary words as candidate target words.

### 3.5 Constructing Multiple Choices

We train machine reading comprehension models using the data built so far to generate four choices for each question. Together with the ground-truth (the target word identified above and removed from the human summary), we have five choices/options for each question. In our work, we propose to use three models, Gated-Attention Reader (Hermann et al., 2015), Attentive Model and Attention Model with Word Gloss to generate the candidate options. Please find details of the models in Appendix B and Appendix C as well as the training details in Appendix D.

We adopt the idea of k-fold cross validation to train the above mentioned three models to generate candidate answer words. Specifically, we split the data into 4 folds. Each time, we train the baseline models on 3 folds of data and use the trained

	MRR	R@1	R@5	R@10
GARReader	0.245	0.175	0.314	0.378
AttReader	0.235	0.167	0.300	0.363
+gloss	0.179	0.123	0.227	0.276

Table 3: Three baseline models are used to generate candidate multiple choices for Subtask 1. The table shows their performance on the XSum dataset, evaluated with MRR (Craswell, 2009), Recall@1, Recall@5, and Recall@10.

	MRR	R@1	R@5	R@10
GARReader	0.343	0.268	0.422	0.484
AttReader	0.348	0.273	0.424	0.490
+gloss	0.228	0.166	0.286	0.345

Table 4: Three baseline models are used to generate candidate multiple choices for Subtask 2. The table shows their performance on the XSum dataset, evaluated with MRR, Recall@1, Recall@5, and Recall@10.

models to predict candidate words on the remaining 1-fold data. With 4-fold iteration, we obtain predication of each model on the entire data. The performance of the three baseline models are listed in Table 3 for Subtask 1 and Table 4 for Subtask 2, using several typical retrieval-based evaluation metrics.

For each target word that has been removed from the corresponding summary sentence (again, a question is a summary sentence containing a removed target word), we collect top-10 words predicted by each of the three models. In this way, we can collect a candidate word pool of 30 predicted word tokens for each removed target word. To avoid including multiple correct choices for each question, we adopt synonym and context similarity filtering methods described in Section 3.4. Specifically we first calculate similarity between the ground-truth target word and each word type in the pool. We exclude a word type from the multiple choices if its similarity to the ground-truth is higher than 0.85. In addition, we also exclude synonyms of the ground-truth target word. For the remaining word tokens in the pool, we select four most frequent word types (a word type may have multiple tokens in the pool). Together with the ground-truth word, we obtain five choices for each question.

### 3.6 Further Quality Control

We further make the following efforts to remove noise in the dataset and improve the datasets’ qual-

ity. We observe that up to now, there are mainly two kinds of noise in our dataset: 1) some target words cannot be inferred solely based on the corresponding passage; 2) more than one of the multiple choices are correct answers.

The first issue is mainly related to the property of the XSum dataset, in which the first sentence of a passage is used as the summary. The second type of problems are often caused by our automatic generation method. Although we have applied strict rules in Section 3.4 to handle this, among a small portion of the resulting data, multiple potentially correct answers still exist in candidate answers.

To further ensure the quality of our dataset, we invite workers in Amazon Mechanical Turk to perform further data selection. Each annotator needs to follow the procedure of Appendix A to answer the question and annotate relevant information, with which further data selection is applied. To ensure quality, we only include workers from English-speaking countries and only if their previous HITs’ approval rates are above 90%. To see more details about this process, please refer to Appendix E.

### 3.7 ReCAM Data Statistics

Table 5 lists the size of our ReCAM datasets, i.e., numbers of questions. For example, in total Subtask 2 has 6,186 questions, which are split into training/development/test subsets.

Dataset	Subtask 1	Subtask 2	Total
Train	3,227	3,318	6,545
Dev	837	851	1,688
Test	2,025	2,017	4,042
Total	6,089	6,186	12,275

Table 5: Size of the ReCAM Dataset.

## 4 Systems and Results

Our shared task received 23 submissions to Subtask 1, 28 submissions to Subtask 2, and 29 submissions to Subtask 3. We use *accuracy* as the evaluation metric for the three subtasks.

In general, most participating teams use pre-trained language models in their systems such as BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020), DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2020), XLNet (Yang et al., 2019), T5 (Raffel et al., 2020). Data augmentation, external knowledge resources,

and/or transfer learning are additionally used by many teams to further enhance their model performance.

#### 4.1 Subtask 1: ReCAM-Imperceptibility

Table 6 shows all the official submissions and most of them outperform the baseline model. The baseline used for Subtask 1 is the Gated-Attention (GA) Reader (Dhingra et al., 2017). The GA Reader uses a multi-layer iterated architecture with a gated attention mechanism to derive better query-aware passage representation. The motivation behind using GA Reader is to have a simple comparison between our task and the CNN/Daily Mail reading comprehension dataset since GA Reader achieves reasonably good performance on the CNN/Daily Mail reading comprehension dataset.

Note that the last column of the table lists the accuracy (*Acc. Cross*) for models trained on the Subtask 2 training data and tested on the Subtask 1 testset. We will discuss those results later in Section 4.3.

The best result in Subtask 1 was achieved by team SRC-B-roc (Zhang et al., 2021) with an accuracy of 0.951. The system was built on a pre-trained ELECTRA discriminator and it further applied upper attention and auto-denoising mechanism to process long sequences. The second-placed system, PINGAN omini-Sinitic (Wang et al., 2021), adopted an ensemble of ELECTRA-based models with task-adaptive pre-training and a multi-head attention based multiple-choice classifier. ECNU-ICA-1 (Liu et al., 2021) ranked third in this subtask with a knowledge-enhanced Graph Attention Network and a semantic space transformation strategy.

Most teams in Subtask 1 utilize pre-trained language models (PLM), like BERT (Devlin et al., 2019), ALBERT (Lan et al., 2020), DistilBERT (Sanh et al., 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark et al., 2020), DeBERTa (He et al., 2020), XLNet (Yang et al., 2019), T5 (Raffel et al., 2020). SRC-B-roc (Zhang et al., 2021) conducted an ablation study regarding the performance discrepancy of different transformers-based pre-training models. They tested BERT, ALBERT, and ELECTRA by directly fine-tuning the pre-trained LMs on the ReCAM data. ELECTRA outperforms BERT and ALBERT by large margins, which may be due to the different learning objec-

Rank	Team	Acc	Acc. Cross
-	GA Reader	25.1	-
1	SRC-B-roc	95.1	91.8 (↓ 3.3)
2	PINGAN-Omini-Sinitic	93.0	91.7 (↓ 1.3)
3	ECNU-ICA-1	90.5	88.6(↓ 1.9)
4	tt123	90.0	86.2(↓ 3.8)
5	cxn	88.7	-
6	nxc	88.6	74.2(↓ 14.4)
7	ZJUKLAB	87.9	-
8	IIE-NLP-Eyas	87.5	82.1(↓ 5.4)
9	hzxx1997	86.7	-
10	XRJL	86.7	81.8(↓ 4.9)
11	noobs	86.2	78.6(↓ 7.6)
12	godrevl	83.1	-
13	ReCAM@IITK	82.1	80.7(↓ 1.4)
14	DeepBlueAI	81.8	76.3(↓ 5.5)
15	LRG	75.3	61.8(↓ 13.5)
16	xuliang	74.7	-
17	Llf1206571288	72.8	-
18	Qing	71.4	-
19	NEUer	56.6	51.8(↓ 4.8)
20	CCLAB	46.3	35.2(↓ 11.1)
21	UoR	42.0	39.4(↓ 2.6)
22	munia	19.3	-
23	BaoShanCollege	19.0	-

Table 6: Official results of Subtask 1 and Subtask 3. *Acc* is the accuracy of the models trained on the Subtask 1 training data and tested on the Subtask 1 testset. *Acc. cross* is the accuracy of models trained on the Subtask 2 training data and tested on the Subtask 1 testset.

tives of these pre-trained models.

Most participating systems performed intermediate task pre-training (Pruksachatkun et al., 2020) for their language models. For example, CNN/Daily Mail dataset was selected by ZJUKLAB (Xie et al., 2021a) to further pre-train their language models. The CNN/Daily Mail dataset and Newsroom dataset boost model performance on both Subtask 1 and Subtask 2. Data augmentation methods are also popular among participants. ZJUKLAB (Xie et al., 2021a) performed negative data augmentation with a

language model to leverage misleading words. IIE-NLP-Eyas (Xie et al., 2021b) adopted template-based input reconstruction methods to augment their dataset and further fine-tuned their language models based on the dataset.

Most teams also used an ensemble of multiple pre-trained language models to further enhance model performance. SRC-B-roc (Zhang et al., 2021) applied Wrong Answer Ensemble (Kim and Fung, 2020) by training the model to learn the correct and wrong answer separately and ensembled them to obtain the final predictions. Stochastic Weight Averaging (Izmailov et al., 2018) was also performed across multiple checkpoints in the same run to achieve better generalization.

In addition, some interesting approaches were additionally used to tackle the task from different perspectives. PINGAN omini-Sinitic (Wang et al., 2021) turned the original multi-choice task into a masked-sentence classification task by adding each option to the placeholder. Noise detection methods and auto denoising methods were further proposed by adding a noise-tolerant loss. ZJUKLAB (Xie et al., 2021a) used label smoothing to encourage the activations of the penultimate layer. ECNU-ICA-1 (Liu et al., 2021) utilized a semantic space transformation strategy to convert ordinary semantic representations into abstract representations for classification.

Many teams used external knowledge resources to further improve model performance. WordNet (Fellbaum, 1998) was widely used to provide candidate word definitions. ECNU-ICA-1 (Liu et al., 2021) also used ConceptNet5 (Speer et al., 2016) and Graph Neural Network in their systems. To alleviate the noise induced by incorporating structured knowledge through unimportant edges, they propose a noise reduction strategy. owlmx used the MRC Psycholinguistic Database to obtain a measurement of *imperceptibility* abstractness.

Different pre-processing techniques were proposed in multiple systems. ZJUKLAB (Xie et al., 2021a) used a sliding window to limit input length in training. PINGAN Omini-Sinitic (Wang et al., 2021) used the cycle noisy label detection algorithm to make models more robust.

Much interesting analysis regarding the failure cases and data distribution was discussed in several system description papers. XRJL (Jiang et al., 2021) found that for a few questions, common

Rank	Team	Acc.	Acc. Cross
-	GA Reader	24.3	-
1	PINGAN-Omini-Sinitic	95.3	94.2 (↓ 1.1)
2	SRC-B-roc	94.9	93.9(↓ 1.0)
3	tt123	93.4	85.8(↓ 7.6)
4	ECNU-ICA-1	93.0	92.8(↓ 0.2)
5	cxn	92.9	-
6	ZJUKLAB	92.8	-
7	nxc	92.7	-
8	hzxx1997	90.2	-
9	XRJL	90.0	87.6(↓ 2.4)
10	IIE-NLP-Eyas	89.6	84.1(↓ 5.5)
11	ReCAM@IITK	87.6	85.2(↓ 2.4)
12	noobs	87.1	82.4(↓ 4.7)
13	DeepBlueAI	86.2	80.7(↓ 5.5)
14	xuliang	81.0	-
15	LRG	77.8	65.6(↓ 12.2)
16	Yotta	71.6	-
17	sayazzad	68.3	-
18	itanhisada	67.7	-
19	NEUer	66.9	45.0(↓ 21.9)
20	YaA@JUST	66.1	-
21	NLP-IIS@UT	64.4	-
22	CCLAB	48.1	31.8(↓ 16.3)
23	K-FUT	47.6	-
24	owlmx	44.8	31.0(↓ 13.8)
25	UIT-ISE-NLP	42.0	27.3(↓ 14.7)
26	UoR	39.1	34.2(↓ 4.9)
27	Noor	19.9	-
28	BaoShanCollege	17.6	-

Table 7: Official results of Subtask 2 and Subtask 3. *Acc* is the accuracy (%) of the models trained on the Subtask 2 training data and tested on the Subtask 2 testset. *Acc. Cross* is the accuracy(%) of models trained on the Subtask 1 training data and tested on the Subtask 2 testset.

sense knowledge was further needed to help find the answer. They also pointed out that there were still a few questions in which multiple candidate choices may serve as appropriate answers.

## 4.2 Subtask 2: ReCAM-Nonspecificity

In Subtask 2, we received 28 submissions. Table 7 shows the official leaderboard. The best result in Subtask 2 was achieved by team PINGAN-omni-Sinitic (Wang et al., 2021) with an accuracy of 0.953, using a model similar to the team’s model in Subtask 1. The second-placed team SRC-B-roc (Zhang et al., 2021) also adopted the same model it used in Subtask 1 with a data augmentation method based on the hypernym hierarchy in WordNet.

In general, the participating teams in Subtask 2 used pre-trained language models and neural networks similar to those they used in Subtask 1. The main differences lie in how the participants performed data augmentation and leveraged external knowledge. For example, in addition to SRC-B-roc (Zhang et al., 2021), the IRG team (Sharma et al., 2021) also performed data augmentation using hypernyms from WordNet.

## 4.3 Subtask 3: Cross-task Performance

In this section, we explore models’ performance across the two types of definitions of abstractness. Specifically, in this subtask, participants train their models on the training set of one subtask and test on the testset of the other subtask. We received 29 submissions in total from the participants.

**Cross-task performance: Subtask 2-to-1 testing.** We asked participants to test their models trained on the Subtask 2 training data on the Subtask 1 test data. The results are shown in the last column of Table 6.

The results we received show that the performance of all systems drops substantially. For some systems ranking among top 10, the accuracy can decrease by 5 points (IIE-NLP-Eyas (Xie et al., 2021b) and XRJL (Jiang et al., 2021)), or even more (14 points for nxc). Some systems show good generalization ability in this Subtask 2-to-1 scenario; the performance of PINGAN-Omini-Sinitic (Wang et al., 2021) is only 1.3 point less, which may be due to the the data augmentation and task adaptive training used in the model.

**Cross-task Performance: Subtask 1-to-2 Testing.** Participants are asked to test their Subtask 1 systems on the Subtask 2 testset. Details of the results can be seen in the last column of Table 7. All systems’ performances drop. For example, among

the top-10 systems, the accuracy decreases by 5 points (IIE-NLP-Eyas (Xie et al., 2021b)) or 7 points (tt123).

However, ECNU-ICA-1 (Liu et al., 2021) shows a very good generalization ability in Subtask 1-to-2 testing. PINGAN-Omini-Sinitic (Wang et al., 2021), SRC-B-roc (Zhang et al., 2021) and XRJL (Jiang et al., 2021)’s systems are rather consistent in this Subtask 1-to-2 cross testing. Some algorithms they used may explain the models’ good generalization ability. ECNU-ICA-1’s algorithm of using knowledge-enhanced Graph Attention Network can provide external knowledge to the model. The Wrong Answer Ensemble algorithm (Kim and Fung, 2020) used in PINGAN-Omini-Sinitic (Wang et al., 2021) is a relatively simple but an effective way of improving model performance and generalization ability. Also, the Stochastic Weight Averaging algorithm across multiple checkpoints is effective for better generalization. XRJL (Jiang et al., 2021) retrieves the definitions of candidate answers from WordNet and feeds them to the model as extra inputs. We also think data augmentation methods contribute to the generalization ability.

## 5 Related Work

There have been tasks being proposed to evaluate machines’ ability on reading comprehension, which either require models to find an entity or text span from the source document as the answer (Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016; Rajpurkar et al., 2016; Trischler et al., 2017), or further generate an answer (Nguyen et al., 2016; He et al., 2018; Kočiský et al., 2018). The cloze-style MRC tasks (Hermann et al., 2015; Onishi et al., 2016; Hill et al., 2016) are most similar to ours, in which the missing words in the cloze questions are entities appearing in source documents. Unlike previous work, ReCAM questions specifically focus on abstract words unseen in the corresponding source documents.

In general, multi-choice questions have been widely used as a tool for language examination to test both humans and machines. In this paper, we follow the multiple-choice framework for our proposed ReCAM task to evaluate computers’ ability in comprehending abstract concepts, in which computers are asked to predict the missing abstract words in human-written summaries.

## 6 Summary

This shared task aims to study the ability of machines in representing and understanding abstract concepts, based on two definitions of abstractness, the *imperceptibility* and *nonspecificity*, in a specific machine reading comprehension setup. We provide three subtasks to evaluate models' ability in comprehending the two types of abstract meaning as well as their generalizability. In Subtask 1, the top system achieves an accuracy of 0.951, and in Subtask 2, an accuracy of 0.953, suggesting the current systems perform well in the specific setup of our share task. In Subtask 3, we found that in general the models' performances dropped in both Subtask 2-to-1 and Subtask 1-to-2 testing. However, some models generalize well, benefiting from technologies such as data augmentation and task adaptive training. We hope the shared task can help shed some light on modelling abstract concepts and help design more challenging tasks in the future.

## References

- Satanjeev Banerjee and Ted Pedersen. 2002. [An adapted lesk algorithm for word sense disambiguation using wordnet](#). In *Computational Linguistics and Intelligent Text Processing, Third International Conference, CICLing 2002, Mexico City, Mexico, February 17-23, 2002, Proceedings*, volume 2276 of *Lecture Notes in Computer Science*, pages 136–145. Springer.
- Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.
- Mark A. Changizi. 2008. [Economically organized hierarchies in wordnet and the oxford english dictionary](#). *Cogn. Syst. Res.*, 9(3):214–228.
- Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [ELECTRA: pre-training text encoders as discriminators rather than generators](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Max Coltheart. 1981. The MRC psycholinguistic database. *The Quarterly Journal of Experimental Psychology*, 33(4):497–505.
- Nick Craswell. 2009. Mean reciprocal rank. In *Encyclopedia of Database Systems*, pages 1703–1703. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. [Gated-attention readers for text comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846. Association for Computational Linguistics.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. [DeBERTa: Decoding-enhanced BERT with disentangled attention](#). *CoRR*, abs/2006.03654.
- Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. [Dureader: a chinese machine reading comprehension dataset from real-world applications](#). In *Proceedings of the Workshop on Machine Reading for Question Answering@ACL 2018, Melbourne, Australia, July 19, 2018*, pages 37–46. Association for Computational Linguistics.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching machines to read and comprehend](#). In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. [The goldilocks principle: Reading children's books with explicit memory representations](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. 2018. [Averaging weights leads to wider optima and better generalization](#). In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pages 876–885. AUAI Press.



- Yuxin Jiang, Ziyi Shou, Qijun Wang, Hao Wu, and Fangzhen Lin. 2021. XRJL-HKUST at SemEval-2021 Task 4: Wordnet-enhanced dual multi-head co-attention for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Hyeondey Kim and Pascale Fung. 2020. [Learning to classify the wrong answers for multiple choice question answering \(student abstract\)](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 13843–13844. AAAI Press.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Tomáš Kočiský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association of Computational Linguistics*, 6:317–328.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A lite BERT for self-supervised learning of language representations](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26. ACM.
- Pingsheng Liu, Linlin Wang, Qian Zhao, Hao Chen, Yuxi Feng, Xin Lin, and Liang He. 2021. ECNU\_ICA.1 SemEval-2021 Task 4: Leveraging knowledge-enhanced graph attention networks for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- George Miller. 1998. *WordNet: An electronic lexical database*. MIT press.
- Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Don’t give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1797–1807. Association for Computational Linguistics.
- Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. [MS MARCO: A human generated machine reading comprehension dataset](#). In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 collocated with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.
- Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2230–2235.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. [Intermediate-task transfer learning with pretrained language models: When and why does it work?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5231–5247. Association for Computational Linguistics.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A Python natural language processing toolkit for many human languages](#).
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *J. Mach. Learn. Res.*, 21:140:1–140:67.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [Squad: 100, 000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Abheesht Sharma, Harshit Pandey, Gunjan Chhablani, Yash Bhartiya, and Tirtharaj Dash. 2021. LRG at SemEval-2021 Task 4: Improving reading comprehension with abstract words using augmentation, linguistic features and voting. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Robert Speer, Joshua Chin, and Catherine Havasi. 2016. [Conceptnet 5.5: An open multilingual graph of general knowledge](#). In *AAAI Conference on Artificial Intelligence*.
- Otfried Spreen and Rudolph W. Schulz. 1966. [Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns](#). *Journal of Verbal Learning and Verbal Behavior*, 5(5):459 – 468.
- DL Theijssen, H van Halteren, LWJ Boves, and NHJ Oostdijk. 2011. On the difficulty of making concreteness concrete.
- Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordani, Philip Bachman, and Kaheer Suleman. 2017. [Newsqa: A machine comprehension dataset](#). In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 191–200. Association for Computational Linguistics.
- Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 680–690. Association for Computational Linguistics.
- Ye Wang, Yanmeng Wang, Haijun Zhu, Bo Zeng, Zhenghong Hao, Shaojun Wang, and Jing Xiao. 2021. PINGAN Omini-Sinitic at SemEval-2021 Task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Xin Xie, Xiangnan Chen, Xiang Chen, Yong Wang, Ningyu Zhang, Shumin Deng, and Huajun Chen. 2021a. ZJUKLAB at SemEval-2021 Task 4: Negative augmentation with language model for reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Yuqiang Xie, Luxi Xing, Wei Peng, and Yue Hu. 2021b. IIE-NLP-Eyas at SemEval-2021 Task 4: Enhancing plm for recam with special tokens, re-ranking, siamese encoders and back translation. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.
- Jing Zhang, Yimeng Zhuang, and Yinpei Su. 2021. TAMAMC at SemEval-2021 Task4: Task-adaptive pretraining and multi-head attention for abstract meaning reading comprehension. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

## A Annotation Script

Please read the instruction before starting answering the question (about 2 minutes reading). The instruction will help you better understand the task. Here is a question with one word missing. Please pick one word to complete it.

**Question:** Wales are waiting to discover the **@placeholder** of the knee injury lock Jake Ball suffered in Scarlets ' 24 - 15 win over Treviso on Friday .

- Before reading the passage, please select one word to answer the question.

challenge    duties    extent    causes    future

**Passage:** Ball, 24, limped off towards the end of the Pro12 game at Parc Scarlets, a week before Wales face England in the Six Nations at Twickenham. "Jake took a knock to the knee at the stage when we scored the try," Scarlets head coach Wayne Pivac said. "It was sensible to get him off as well." Pivac added: "I think he'll be all right - he's thrown a bit of ice so we'll see what the medical boys say." Wales coach Warren Gatland released lock Ball and scrum-half Aled Davies to the region as the Welsh squad prepare for their Six Nations showdown with England on 12 March. Media playback is not supported on this device Pivac also revealed Davies, 23, had been struggling with a virus while Scotland forward John Barclay also suffered an injury during the victory over bottom side Treviso. "John just took a stinger and felt he was favouring one side so he was going to let the team down so it was the right decision and he came off but I'm pretty sure he'll be OK," Pivac added. "Aled Davies was crook for the last couple of days with a virus and had to get through the warm-up. He did that and started so he was always going to run out of petrol. "It was a matter of how long he'd last and he did well to get to where he did."

- After reading and understanding the passage, select one word to answer the question.

challenge    duties    extent    causes    future

- If you think there is another answer to this question, please pick your second choice.

Else, please pick the *"No Other Answer"* option.

challenge    duties    extent    causes    future    No Other Answer

- Which part of the passage supports your answer? Copy the text span to the line below:

Paste corresponding text span here: \_\_\_\_\_

- Does the answer summarize or paraphrase the text span?

Summarize    Paraphrase    Others

- Which part of the passage summarize the information included in the answer? Copy the text span to the line below:

Paste corresponding text span here: \_\_\_\_\_

- How difficult is this problem?

Easy    Medium    Difficult

## B Gated-Attention Reader

The Gated-Attention (GA) Reader (Dhingra et al., 2017), the state-of-art model on CNN/Daily Mail reading comprehension dataset (Hermann et al., 2015), is adapted here in our experiments. The GA Reader uses a multi-layer iterated architecture with a gated attention mechanism, which is based on multiplicative interactions between the query embedding and the intermediate states of a recurrent neural network document reader, to derive better query-aware passage representation. To apply GA Reader to our ARC task, we input the news passage  $p$  as the document and the processed summary  $s$  as the query to GA Reader.

Specifically, for an input passage  $p = [p_1, p_2, \dots, p_{l_p}]$  with  $l_p$  words and its corresponding summary  $s = [s_1, s_2, \dots, s_{l_s}]$  with  $l_s$  words, we first derive their corresponding word embedding sequence  $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{l_p}]$  and  $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{l_s}]$  respectively. Then the GA Reader accepts the  $\mathbf{P}$  and  $\mathbf{S}$  as inputs and return the hidden states  $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, \dots, \mathbf{h}_{l_p}^p]$  and  $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, \dots, \mathbf{h}_{l_s}^s]$  as the sequential representation for passage  $p$  and summary  $s$  respectively. As for the final prediction process, we do not adopt the operations in Dhingra et al. (2017) because in ARC the answer words are unseen in the corresponding passage, however, GA Reader in Dhingra et al. (2017) tries to select a entity word in the passage as the final prediction since their target answer word appears in the passage. So we redesign the part of prediction.

First, the corresponding representation of “@placeholder” in  $\mathbf{H}_s$ , denoted as  $\mathbf{h}_q^s$  ( $q$  is the position index of @placeholder in summary  $s$ ), is used as the final vector representation for summary  $s$ . For the final vector representation  $\mathbf{p}$  for passage  $p$ , a bilinear attention between  $\mathbf{h}_q^s$  and  $\mathbf{H}_p$  is used for its derivation:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}^{att} \mathbf{h}_i^p, \forall i \in [1, \dots, l_p] \quad (1)$$

$$\mathbf{p} = \sum_{i=1}^{l_p} \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j} \mathbf{h}_i^p, \quad (2)$$

We set a token embedding  $\mathbf{a}_t^e$  for each candidate abstractive word  $a_t$  ( $t \in [1, \dots, n_c]$ ,  $n_c$  is the size of candidate set). We first concatenate the  $\mathbf{h}_p^s$  and  $\mathbf{p}$ , then use the bilinear product and softmax to predict the probability distribution over all  $n_c$  candidate

abstractive words.

$$r_t = [\mathbf{h}_q^s; \mathbf{p}]^T \mathbf{W}_p \mathbf{a}_t^e, \forall t \in [1, \dots, n_c], \quad (3)$$

$$o_t = \text{softmax}_t(r_t), \forall t \in [1, \dots, n_c] \quad (4)$$

in which  $o_t$  represents the probability of predicting the candidate abstractive word  $a_t$  as the final answer.

## C Attentive Model

The word gloss, which defines a word sense meaning, has been mainly used in word sense disambiguation (WSD) task and its variants (Lesk, 1986; Moro et al., 2014). Since the goal of ARC is to predict a word that can summarize corresponding information from the source passage, which is an abstracting process, it may be helpful when the gloss, i.e., interpretation of candidate abstractive words, are provided.

We design an attentive model with word gloss (AMWG) as Figure 1 shows. Specifically, all the encoders are 1-layer bi-directional recurrent neural networks (RNNs) with Gated Recurrent Units (GRU) (Cho et al.). For an input news passage  $p = [p_1, p_2, \dots, p_{l_p}]$  with  $l_p$  words, we can derive its hidden states  $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, \dots, \mathbf{h}_{l_p}^p]$  by sending its word embedding sequence  $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{l_p}]$  to the *Passage Encoder*. Similarly, we can derive hidden states  $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, \dots, \mathbf{h}_{l_s}^s]$  for summary  $s$  by inputting its word embedding sequence  $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_{l_s}]$  into the *Summary Encoder* and hidden states  $\mathbf{H}_{g_t} = [\mathbf{h}_1^{g_t}, \mathbf{h}_2^{g_t}, \dots, \mathbf{h}_{l_{g_t}}^{g_t}]$  for gloss  $g_t$  of the candidate word  $a_t$  by sending its word embedding sequence  $\mathbf{G}_t = [\mathbf{g}_1^t, \mathbf{g}_2^t, \dots, \mathbf{g}_{l_{g_t}}^t]$  to the *WordGloss Encoder*.

Similar to Section B, the corresponding representation of “@placeholder”, i.e.,  $\mathbf{h}_q^s$ , is used as the final vector representation for summary  $s$ . And an bilinear attention  $f_{att}^p(\bullet)$  is applied to  $\mathbf{h}_q^s$  and  $\mathbf{H}_p$  as follows:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}_{att}^p \mathbf{h}_i^p, \forall i \in [1, \dots, l_p] \quad (5)$$

$$\alpha_i = \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j}, \forall i \in [1, \dots, l_p] \quad (6)$$

Then  $\mathbf{p}$  is derived as the vector representation for passage  $p$  by the weighed sum of  $\mathbf{H}_p$ , which is further concatenated with the  $\mathbf{h}_q^s$  to form the final summarization vector  $\mathbf{v}$ :

$$\mathbf{p} = \sum_{i=1}^{l_p} \alpha_i \mathbf{h}_i^p, \quad (7)$$

$$\mathbf{v} = \text{concat}(\mathbf{p}, \mathbf{h}_q^s), \quad (8)$$

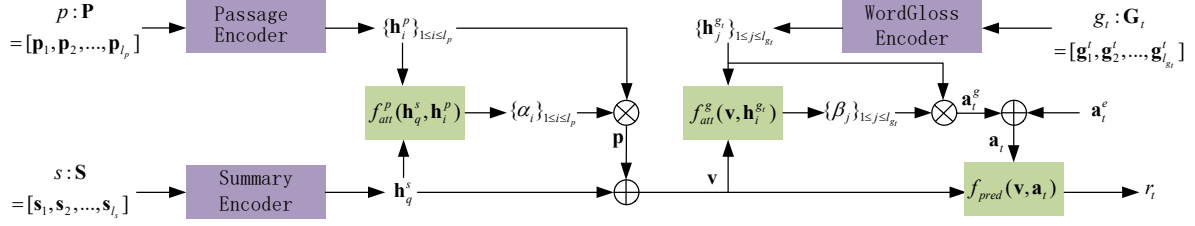


Figure 1: The model architecture of the attentive model with word gloss (AMWG) implemented in this paper.  $\oplus$  denotes the concatenation of input vectors. All the encoders are 1-layer bi-directional GRU-RNNs,  $\otimes$  denotes the weighted sum of vectors.

Another attention  $f_{att}^g(\bullet)$  is applied to  $\mathbf{v}$  and  $\mathbf{H}_{g_t}$ ,

$$e_j = \tanh(\mathbf{W}_{att}^g \mathbf{v} + \mathbf{b})^T \mathbf{h}_j^{g_t}, \forall j \in [1, \dots, l_{g_t}] \quad (9)$$

$$\beta_j = \frac{\exp e_j}{\sum_{i=1}^{l_{g_t}} \exp e_i}, \forall j \in [1, \dots, l_{g_t}], \quad (10)$$

The following weighted sum of  $\mathbf{H}_{g_t}$ , i.e.  $\mathbf{a}_t^g$ , is derived as the final vector representation for the gloss of candidate word  $a_t$ :

$$\mathbf{a}_t^g = \sum_{j=1}^{l_{g_t}} \beta_j \mathbf{h}_j^{g_t} \quad (11)$$

We also set a token embedding  $\mathbf{a}_t^e$  for each candidate word  $a_t$  ( $t \in [1, \dots, n_c]$ ,  $n_c$  is the size of candidate set), which is further concatenated with  $\mathbf{a}_t^g$  to build the final representation  $\mathbf{a}_t$  for candidate word  $a_t$ . For the final prediction, we input the summarization vector  $\mathbf{v}$  and candidate representation vector  $\mathbf{a}_t$  to  $f_{pred}(\bullet)$  and apply the softmax to derive the probability distribution over all  $n_c$  candidate abstractive words,

$$\mathbf{a}_t = \text{concat}(\mathbf{a}_t^g, \mathbf{a}_t^e), \quad (12)$$

$$r_t = \mathbf{v}^T \mathbf{W}_{pred} \mathbf{a}_t, \forall t \in [1, \dots, n_c], \quad (13)$$

$$o_t = \text{softmax}_t(r_t), \forall t \in [1, \dots, n_c] \quad (14)$$

in which  $o_t$  gives the probability of predicting the candidate word  $a_t$  as the final answer. The word gloss, which defines a word sense meaning, has been mainly used in word sense disambiguation (WSD) task and its variants (Lesk, 1986; Moro et al., 2014). Since the goal of ARC is to predict a word that can summarize corresponding information from the source passage, which is an abstracting process, it may be helpful when the gloss, i.e., interpretation of candidate abstractive words, are provided.

We design an attentive model with word gloss (AMWG) as Figure 1 shows. Specifically, all the encoders are 1-layer bi-directional recurrent neural networks (RNNs) with Gated Recurrent Units (GRU) (Cho et al.). For an input news passage  $p = [p_1, p_2, \dots, p_{l_p}]$  with  $l_p$  words, we can derive its hidden states  $\mathbf{H}_p = [\mathbf{h}_1^p, \mathbf{h}_2^p, \dots, \mathbf{h}_{l_p}^p]$  by sending its word embedding sequence  $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{l_p}]$  to the *Passage Encoder*. Similarly, we can derive hidden states  $\mathbf{H}_s = [\mathbf{h}_1^s, \mathbf{h}_2^s, \dots, \mathbf{h}_{l_s}^s]$  for summary  $s$  by inputting its word embedding sequence  $\mathbf{S} = [s_1, s_2, \dots, s_{l_s}]$  into the *Summary Encoder* and hidden states  $\mathbf{H}_{g_t} = [\mathbf{h}_1^{g_t}, \mathbf{h}_2^{g_t}, \dots, \mathbf{h}_{l_{g_t}}^{g_t}]$  for gloss  $g_t$  of the candidate word  $a_t$  by sending its word embedding sequence  $\mathbf{G}_t = [g_1^t, g_2^t, \dots, g_{l_{g_t}}^t]$  to the *WordGloss Encoder*.

Similar to Section B, the corresponding representation of “@placeholder”, i.e.,  $\mathbf{h}_q^s$ , is used as the final vector representation for summary  $s$ . And an bilinear attention  $f_{att}^p(\bullet)$  is applied to  $\mathbf{h}_q^s$  and  $\mathbf{H}_p$  as follows:

$$e_i = \mathbf{h}_q^{sT} \mathbf{W}_{att}^p \mathbf{h}_i^p, \forall i \in [1, \dots, l_p] \quad (15)$$

$$\alpha_i = \frac{\exp e_i}{\sum_{j=1}^{l_p} \exp e_j}, \forall i \in [1, \dots, l_p] \quad (16)$$

Then  $\mathbf{p}$  is derived as the vector representation for passage  $p$  by the weighed sum of  $\mathbf{H}_p$ , which is further concatenated with the  $\mathbf{h}_q^s$  to form the final summarization vector  $\mathbf{v}$ :

$$\mathbf{p} = \sum_{i=1}^{l_p} \alpha_i \mathbf{h}_i^p, \quad (17)$$

$$\mathbf{v} = \text{concat}(\mathbf{p}, \mathbf{h}_q^s), \quad (18)$$

Another attention  $f_{att}^g(\bullet)$  is applied to  $\mathbf{v}$  and  $\mathbf{H}_{g_t}$ ,

$$e_j = \tanh(\mathbf{W}_{att}^g \mathbf{v} + \mathbf{b})^T \mathbf{h}_j^{g_t}, \forall j \in [1, \dots, l_{g_t}] \quad (19)$$

$$\beta_j = \frac{\exp e_j}{\sum_{i=1}^{l_{g_t}} \exp e_i}, \forall j \in [1, \dots, l_{g_t}], \quad (20)$$

The following weighted sum of  $\mathbf{H}_{g_t}$ , i.e.  $\mathbf{a}_t^g$ , is derived as the final vector representation for the gloss of candidate word  $a_t$ :

$$\mathbf{a}_t^g = \sum_{j=1}^{l_{g_t}} \beta_j \mathbf{h}_j^{g_t} \quad (21)$$

We also set a token embedding  $\mathbf{a}_t^e$  for each candidate word  $a_t$  ( $t \in [1, \dots, n_c]$ ,  $n_c$  is the size of candidate set), which is further concatenated with  $\mathbf{a}_t^g$  to build the final representation  $\mathbf{a}_t$  for candidate word  $a_t$ . For the final prediction, we input the summarization vector  $\mathbf{v}$  and candidate representation vector  $\mathbf{a}_t$  to  $f_{pred}(\bullet)$  and apply the softmax to derive the probability distribution over all  $n_c$  candidate abstractive words,

$$\mathbf{a}_t = \text{concat}(\mathbf{a}_t^g, \mathbf{a}_t^e), \quad (22)$$

$$r_t = \mathbf{v}^T \mathbf{W}_{pred} \mathbf{a}_t, \forall t \in [1, \dots, n_c], \quad (23)$$

$$o_t = \text{softmax}_t(r_t), \forall t \in [1, \dots, n_c] \quad (24)$$

in which  $o_t$  gives the probability of predicting the candidate word  $a_t$  as the final answer.

## D Training Details

We train all models using the non-negative log-likelihood as the objective function. The gloss of candidate words are derived from *WordNet* using the NLTK tools (Bird and Loper, 2004). Specifically, we first lemmatize the candidate word and use the lemmatized word as the query word for the searching in *WordNet*. To cope with the semantic ambiguity of words, we just concatenate the gloss of the first sense in each retrieved POS for the query word with corresponding POS tag as the delimiter.

Models in our experiments are trained with the following hyperparameter settings: All word embeddings and token embeddings  $\mathbf{a}_t^e$  have 300 dimensions and are initialized with Glove (Pennington et al., 2014). The passage  $p$  and summary  $s$  share one set of word embeddings, which are fixed during training. The glosses  $\{g_t\}$  for candidate words  $\{a_t\}$  keep its own word embeddings.

The hidden state vectors of all bi-directional GRU-RNNs in all models have 150 dimensions. The number of attention hops in GA Reader is set to 3. The batch size is set to 32. The method of Adam (Kingma and Ba, 2015) is adopted for optimization with initial learning rate  $1e - 03$ . A dropout with rate 0.3 is applied to the input layers for all GRU-RNN encoders and the final summarization vector  $\mathbf{v}$ .

## E Annotation Selection

To ensure most of our annotation is valid, we select annotations satisfying the following criteria: a) the average accuracy is higher than 40%; b) both text spans should not be empty; c) if the difficulty level is rated as easy, then this data sample should be answered correctly.

# TA-MAMC at SemEval-2021 Task 4: Task-adaptive Pretraining and Multi-head Attention for Abstract Meaning Reading Comprehension

Jing Zhang, Yimeng Zhuang, Yinpei Su\*

Samsung Research China-Beijing (SRC-B)

{jing97.zhang, ym.zhuang, yinpei.su}@samsung.com

## Abstract

This paper describes our system used in the SemEval-2021 Task 4 Reading Comprehension of Abstract Meaning, achieving 1st for subtask 1 and 2nd for subtask 2 on the leaderboard. We propose an ensemble of ELECTRA-based models with task-adaptive pretraining and a multi-head attention multiple-choice classifier on top of the pre-trained model. The main contributions of our system are 1) revealing the performance discrepancy of different transformer-based pretraining models on the downstream task, 2) presentation of an efficient method to generate large task-adaptive corpora for pretraining. We also investigated several pretraining strategies and contrastive learning objectives. Our system achieves a test accuracy of 95.11 and 94.89 on subtask 1 and subtask 2 respectively.

## 1 Introduction

Machine reading comprehension (MRC) is one of the key tasks for measuring machines' ability of understanding human languages and reasoning, it can be used broadly in real world applications such as Q&A systems and dialogue systems. MRC often comes in a triplet style  $\{passage, question, answer\}$ , given a context passage, questions related with this passage is asked, and the machine is expected to give the answers. The question-answer form can be question-answer pair, where the answer text is to be provided by machines, or statement form where the answer is to be filled in as cloze or multiple choices selection. By the type of answer formation, MRC can be divided into extractive and generative MRC, the former takes segments from the passage as the answer and the latter requires answer text generation based on the understanding of the passage.

Contribution during Internship in Samsung Research China-Beijing.

Generative MRC is harder than extractive MRC, since it requires more on information integration and reasoning besides focusing on relevant information.

One of the classic MRC approach focuses on matching networks, various network structures have been proposed to capture the semantic interaction within passages/questions/answers. Recent years, pre-trained language models (LMs) have brought non-trivial progress to the performance on MRC, and there's a decline of complex matching networks (Zhang et al., 2020). Plugging matching networks on top of pre-trained LMs can see either improvements or degradation in performance (Zhang et al., 2020; Zhu et al., 2020). Multiple-choice MRC (MMRC) often lacks abundant training data for deep neural networks (this might be caused by the expensive human labelling cost) and it results in a limitation to take full advantage of the pre-trained LMs.

The SemEval-2021 task 4 Reading Comprehension of Abstract Meaning (Zheng et al., 2021), is a multiple-choice English MRC task, aiming at investigating the machine's ability to understand abstract concepts in two aspects: subtask 1, non-concrete concepts, e.g. service/economy compared with trees/red; subtask 2, generalized/summarized concepts, like vertebrate compared with monkey.

We propose an approach based on the pre-trained LM ELECTRA (Clark et al., 2020), with an ensemble of multi-head attention (Vaswani et al., 2017) multiple-choice classifier, and WAE (Kim and Fung, 2020) to get the final prediction. **First**, we conduct task-adaptive pretraining, which is transfer learning using in-domain data on the ELECTRA model. **Then** we fine-tune the ReCAM task using a multi-head attention multiple choice classifier (MAMC) on top of the ELECTRA model. **Finally** we enhance the system with WAE and ensemble them all to get the best generalization capability.

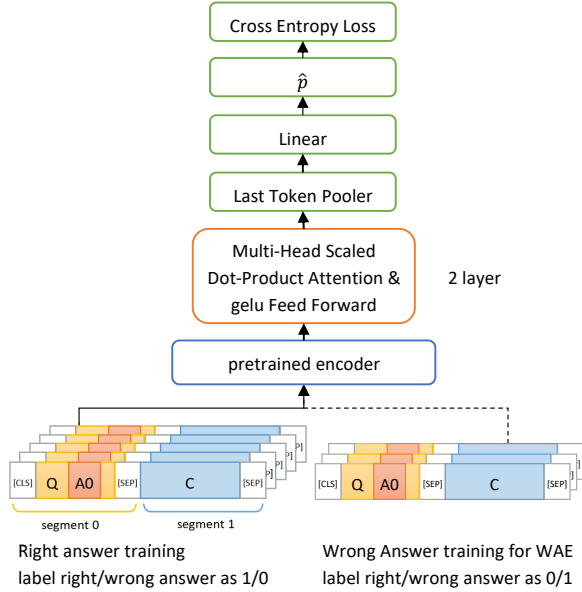


Figure 1: The overall architecture of our proposed system

In addition, we also investigated into transfer learning with natural language inference (NLI) tasks and contrastive learning objectives.

## 2 System Overview

Figure 1 illustrates the overall architecture of our system. The options are substituted into the query to form a complete context, rather than separate query/option segments, in order to get a less semantically ambiguous representation of the query and option. The option-filled query and context tokens are concatenated as in Figure 1, wrapped by [CLS] token and [SEP] tokens. Token embeddings are added up with segment embeddings and positional encodings to form the input for the pre-trained encoder. Then the representations from the encoder are put through a multi-head attention multiple choice classifier, which consists of 1) a 2 layer multi-head attention feed forward network to further capture the task specific query-context interactions, 2) a pooler and a linear transformation to get the final cross entropy loss. We first conduct task-adaptive pretraining on the system, and then fine-tune on the ReCAM dataset, the final model is an ensemble model by several generalization techniques including wrong answer ensemble.

### 2.1 Task-adaptive Pretraining

Pre-trained LMs and their downstream applications have definitely proved the power of transfer learning. The precondition of transfer learning is that the pretraining tasks have shared underlying sta-

tistical features with downstream tasks. Usually in-domain data brings more improvement on downstream tasks than out-of-domain data (Sun et al., 2019; Gururangan et al., 2020).

The genre of the ReCAM task dataset is news (confirmed by manual random checking), we argue that the task of news abstractive summarization provides high quality further pretraining dataset for ReCAM. The dataset comes in  $\{article, summary\}$  pairs, the articles are crawled from formal online news publishers and the summaries are generated by humans and contain abstractive key information of the articles. News abstractive summarization aims at teaching machines to grasp the key information of the whole context by letting machines to generate the summary text.

We regenerate the ReCAM style multiple-choice dataset from the original news abstractive summarization dataset. Letting the article/summary be the passage/question, the regeneration strategy mainly includes 2 steps: 1) identify the abstract concepts in the news dataset, 2) generate gold and pseudo options. In step 1, we count the part-of-speech (POS) tags of all gold labels on the ReCAM training data as shown in Figure 2 (nouns, adjectives and adverbs are the most frequent option tags), and use a similar POS tag distribution to randomly sample word in the summary text that does not appear in the corresponding news article as gold option. In step 2, the gold option in the summary is replaced by the mask token and fed into the pre-trained LM. The LM predicts the mask token and we select some of the top ranking ones as pseudo options. Specifically, setting a high ranking threshold (e.g. top 5) would get words too similar with the gold option, which would bring extra ambiguity to the model, some relaxation on the ranking threshold would ease the problem. This method is automatic, cheap to apply on large dataset, while the abstract concept approximation in step 1 would bring some noise, such as person’s names and geolocations are sometimes selected, but by our experiment result the overall pretraining performance is not hurt, the noisy samples should account for a small fraction.

In addition, it is reported that NLI task transfer

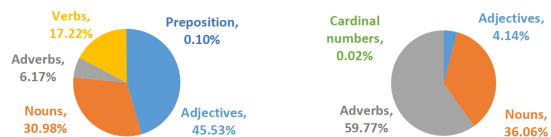


Figure 2: Subtask 1 (left) and subtask 2 (right) gold options POS Tag distribution



Dataset	# Passages	avg. doc len		avg. qry/smry len	
	training/dev/test	# words	# sent.	# words	# sent.
ReCAM subtask 1	3227/837/-	302.15	13.1	24.69	1
ReCAM subtask 2	3318/851/-	481.51	21.08	26.9	1
XSUM	20.3k/11.3k/1.1k	431.07	19.77	23.26	1
NEWSROOM	99.5k/-/-	658.6	-	26.7	-

Table 1: ReCAM/XSUM/NEWSROOM datasets statistics

learning performs well in several MMRC tasks (Jin et al., 2020). Therefore we also explored the MNLI (Williams et al., 2017) and RTE (Wang et al., 2018) tasks transfer learning for the ReCAM task, but it results in degradation. This indicates that NLI tasks are not generally fit for further pre-training in MMRC on pre-trained LMs.

## 2.2 Multi-head Attention Multiple Choice Classifier

The classifier takes the last layer hidden representations from the pre-trained encoder, applies the multi-head attention and feed forward non-linearity, each with a layer normalization (Vaswani et al., 2017). After that the last token is pooled, which is selecting the hidden vector from the hidden embeddings by the index of the last [SEP] token in the input, and then linearly transformed to get the probability of each  $\{query_{option\_filled}, context\}$  candidate pair.

In addition, we also explored the contrastive learning objective. When humans do MMRC, they usually compare the options according to the passage, exclude the wrong ones and then analyze further on the indeterminate ones. Inspired by this, we experimented with triplet loss (Weinberger et al., 2006) (among  $\{input_{non\_filled}, input_{gold}, input_{pseudo}\}$ ) and n-tuplet loss (Sohn, 2016) on all option-filled query and context within one sample. However the contrastive learning objective degrades the performance, suggesting these learning objectives are not as suitable for the ReCAM task as the MLE loss.

## 2.3 Wrong Answer Ensemble

Wrong Answer Ensemble (Kim and Fung, 2020) is a relatively simple yet effective method (Zhu et al., 2020). Kim proposed to train the model to learn the correct and the wrong answers separately and ensemble them to get the final prediction. In 2.2, the correct answer is labelled as 1 and wrong as 0 for correct answer training. Wrong answer training does the opposite labelling (correct/wrong answers

as 0/1) and fine tune the model with binary cross entropy loss as below:

$$loss_w = - \sum y \log \hat{y} + (1 - y) \log(1 - \hat{y}) \quad (1)$$

The two models’s output,  $p_c$  and  $p_w$  are linearly combined to give the final prediction. A simple linear regression is leveraged to find the best value of weight  $w$ .

$$\hat{p} = p_c - w \cdot p_w \quad (2)$$

## 3 Experimental Setup

### 3.1 Dataset

We leverage external news abstractive summarization datasets for transfer learning, and then fine tune our model on the ReCAM dataset.

**ReCAM.** Dataset for the SemEval-2021 Task 4, consisting of news articles (verified by manually random checking) and multiple-choice questions.

**XSUM.** XSUM (Narayan et al., 2018) consists of 227k BBC articles from 2010 to 2017 covering a wide variety of subjects along with professionally written single-sentence summaries.

**NEWSROOM.** NEWSROOM (Grusky et al., 2018) is a dataset of 1.3 million news articles and summaries written by authors and editors in newsrooms of 38 major news publications between 1998 and 2017. After a coarse selection (filtering out lengthy articles/summaries, summaries duplicate with news articles, articles with unqualified pseudo options), about 229k article/summary pairs are used.

The data statistics are listed in Table 1, the 3 news datasets share similar article and query lengths.

### 3.2 Training Details

We compare the baseline performance of 3 kinds of Transformer-based models, BERT/ALBERT/ELECTRA, and select ELECTRA as our encoder. We adopt most hyper parameter settings from the ELECTRA large model, specifically our learning rate is  $1e-5$ , batch size is 32 and gradient clip norm

Pre-trained model	subtask 1 dev acc.	subtask 2 dev acc.
BERT base	61.25	58.28
BERT large	66.31	67.33
ALBERT base	50.78	50.29
ALBERT large	80.88	79.08
ELECTRA base	76.82	76.97
ELECTRA large	90.20	90.13

Table 2: Baseline performance of different pre-trained Models

threshold is set to 1. In the task-adaptive data generation process, We set the threshold as top 10 for pseudo options selection, filtering out the word piece predictions(word pieces all start with a ”#” in the vocabulary) and randomly select 4 words as pseudo options. See the appendix for hyperparameter details. Training was done on NVidia V100 GPUs. All the performance data is on the dev set.

## 4 Results

### 4.1 Pre-trained LM Selection and Task-adaptive Pretraining

The baseline performance of BERT, ALBERT and ELECTRA is tested by directly fine-tuning the ReCAM data on the pre-trained LMs. The results are shown in Table 2. ELECTRA outperforms the other two models with large margins. This may be caused by the learning objective difference among the models. The BERT/ALBERT models learn to predict the masked word from the vocabulary, while the ELECTRA model learns to predict whether each of the token in the input is replaced or not, which learns more about unreasonable co-occurrence knowledge besides reasonable co-occurrences and may help in digging deeper implicit semantic relations for ReCAM. Therefore the ELECTRA large model is selected as the encoder for further experiments.

The XSUM/NEWSROOM regenerated data (denoted as XN) is used for in-domain pretraining on the encoder, and the subtask 1 is fine tuned after pretraining. The prediction accuracy grows with more data fed, as shown in Figure 3. In the end of the task-adaptive pretraining, subtask 1 achieves dev accuracy 92.73, 2.80% higher than directly fine-tuning on the encoder, subtask 2 gets 92.95, increased by 3.13%.

Besides the task-adaptive pretraining and fine-tuning, we also tried multitask learning with

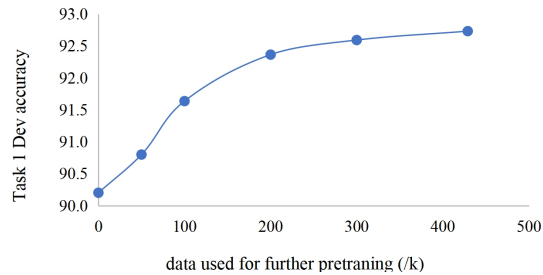


Figure 3: Subtask 1 fine-tuning performance increases with more data for further pretraining

Transfer learning setting	subtask 1	subtask 2
XN	92.73	92.95
ReCAM/ $XN_{multitask}$	92.35	92.36
MNLI	78.14	81.67
RTE	88.53	89.36

Table 3: Dev accuracy for different transfer learning settings

XSUM/NEWSROOM and the ReCAM data together (up sampling the ReCAM data as 3:7 with the news dataset). The results in Table 3 shows that this approach outperforms the encoder baseline, while slightly worse than the full news data pre-trained model, this model is used for ensemble. Using MNLI/RTE for further pretraining hurt the ReCAM fine-tuning performance, especially MNLI pretraining brings about 10% accuracy decrease than the baseline.

### 4.2 On-top Classifier and WAE

Adding MAMC on the top of the encoder helps increase accuracy on the ReCAM subtask 1 and subtask 2, the results are shown in Table 4. Further we applied the WAE to squeeze marginal increases on prediction accuracy. While option contrastive learning (OCL) does not bring performance improvement, worse than directly fine-tuning the encoder with multiple choice classifier.

Settings	subtask 1	subtask 2
Baseline	90.20	90.13
transfer learning	92.73	92.95
+ MAMC	93.64	93.79
+ WAE	93.94	94.07
OCL (triplet loss)	86.38	-
OCL (n-tuple loss)	85.32	-

Table 4: Dev accuracy on different transfer learning settings

Generalization Procedures	subtask1	subtask2
data repar. (3 sets)	93.72 94.01	93.65 94.48
task data aug.	93.82	94.36
	93.29	93.36

Table 5: Dev accuracy of subtask 1/2 over generalization procedures.

### 4.3 Improving Generalization

We mainly applied 3 procedures below for better generalization, and the ensemble of all the models have achieved test accuracy 95.11 on subtask 1 and 94.89 on subtask 2 on the ReCAM leaderboard.

1) Data repartitioning (mix the train/dev sets, and randomly split into new train/dev sets by 8:2 or 9:1) aims to smooth the distribution difference among different train/dev data partition. As is shown in the Table 5, the accuracy of different sets differs, with some higher than then original partition.

2) Augmenting the task data itself for fine-tuning, to mask different word than the original gold option (if there exists) using the method in 2.1. The accuracy remains almost the same after adding the task augmented data. This suggests that our automatic augmentation method makes lower quality samples than the labelling data, while not too noisy that it can contribute to the robustness of the model.

3) We also did Stochastic Weight Averaging (Izmailov et al., 2018) across multiple checkpoints in the same run to get better generalization (SWA dose not improve dev error but test error, so it’s not listed in Table 5).

### 4.4 Fail Cases Analysis

We manually checked and categorized the fail cases on subtask 1 and subtask 2 into 5 classes (given roughly 850 dev cases, the total fail cases is around 50 for both subtask 1 and subtask 2). The detailed examples for each class can be found in the appendix.

- EC0, easy case. In these cases, the answer can be inferred from the query/context, while the model fails to give the correct prediction
- EC1, complicated coreference. Such cases has complicated coreference relations, though the answer can be inferred, the coreferences hinder the model from understanding correctly
- EC2, complex reasoning. In these cases, either the information related with the answer

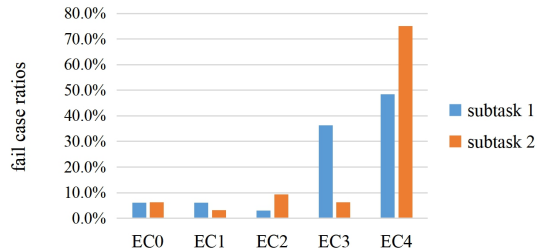


Figure 4: Subtask 1/2 fail case distribution

is sparse in the query/context, or the facets related with the answer is separated with intense unrelated noisy information

- EC3, external knowledge dependency. Only with the external knowledge can one give a correct answer
- EC4, ambiguity in sample cases. This category includes cases for which we think humans are not able to select the correct answer. Either the information is not enough to make a decision or there are more than one reasonable answers.

Figure 4 shows the ratios of each fail case class, the EC4 is the major class, 48.5% for subtask 1 and 75.0% for sutask 2. The following is EC3, 36.4% for subtask 1 and 6.3% for subtask 2. EC0 and EC1 are minor classes among all. With the system backbone being pre-trained LM with a matching network, it’s not a surprise to see EC1 and EC3 failures, while the few EC0 and EC2 failures shows that our system learns well to capture abstract concepts within the query/article pair.

## 5 Conclusion

Our system takes the large pre-trained LM ELECTRA, and enhance it with in-domain transfer learning and a multi-head multiple-choice classifier on top. We compared the benchmark performance of different pre-trained LMs (BERT, ALBERT and ELECTRA) on the SemEval-2021 task 4, the result shows that different pretraining objective/dataset can lead to different inclination of model knowledge and large performance discrepancy on the downstream task. Task-adaptive pretraining has contributed the main improvement, and multi-head multiple-choice classifier and WAE bring marginal improvement. We also investigated into option contrastive learning and multitask learning, the degradation of performance suggests that triplet and n-tuplet contrastive loss is not suitable for this task and NLI is not generally beneficial for MMRC tasks.

## References

- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. *arXiv preprint arXiv:1804.11283*.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.
- Pavel Izmailov, Dmitrii Podoprikin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2018. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.
- Di Jin, Shuyang Gao, Jiun-Yu Kao, Tagyoung Chung, and Dilek Hakkani-tur. 2020. Mmm: Multi-stage multi-task learning for multi-choice reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8010–8017.
- Hyeondey Kim and Pascale Fung. 2020. Learning to classify the wrong answers for multiple choice question answering (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13843–13844.
- Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don’t give me the details, just the summary! *Topic-aware Convolutional Neural Networks for Extreme Summarization*. In.
- Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865.
- Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.
- Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. 2006. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.
- Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020. Machine reading comprehension: The role of contextualized language models and beyond. *arXiv preprint arXiv:2005.06249*.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Duma: Reading comprehension with transposition thinking.

## Appendix

### A Examples for each error case category

<b>EC0</b>	<b>easy case</b>
question	Two men have been arrested on suspicion of murdering a man who died after being pulled out of a fish @placeholder .
passage	The dead man went into cardiac arrest after rescuers responding to reports of a drowning found him in the water off St Michaels Road, Stoke-on-Trent. . .
options	0. term      1. boat      2. shop      3. pool      4. life
<b>EC1</b>	<b>complicated coreference</b>
question	Ceredigion council has failed to co-operate with an investigation into the @placeholder of a Llandysul residential home , a union has claimed
passage	Unison said the council had failed to provide answers for social care expert Tony Garthwaite, heading the investigation, and that he was not able to complete his report. Awel Deg care home was shut in February 2014. . . Awel Deg was closed following the suspension of 11 members. . . would re-open as a dementia home in spring 2015
options	0. creation      1. collapse      2. closure      3. safety      4. fate
<b>EC2</b>	<b>complex reasoning</b>
question	Six British teams @placeholder the draw for the Champions League group stage , which takes place on Thursday at 17:00 BST in Monaco .
passage	Premier League champions Chelsea, runners-up Tottenham and third-placed Manchester City are all in the draw. They will be joined by Europa League winners Manchester United, as well as Liverpool and Scottish champions Celtic who both came through qualifying. The group stages of the competition begin on 12-13 September. The last time six British teams qualified for the group stages was in 2007-08, when English sides Manchester United, Chelsea, Liverpool and Arsenal were joined by Scottish clubs Celtic and Rangers. The final saw Sir Alex Ferguson’s United defeat Avram Grant’s Chelsea on penalties. Scroll to the bottom to see the full list of teams and the pots they are in. . . Match day four: 31 October-1 November Match day five: 21-22 November Match day six: 5-6 December
options	0. announced      1. dominate      2. started      3. await      4. remains
<b>EC3</b>	<b>external knowledge dependency</b>
question	The M4 has been closed westbound near Newport after an overhead @placeholder became loose in high winds .
passage	The carriageway was shut from junction 24 Coldra to 28 at Tredegar Park. Officials said it led to very slow traffic as motorists were forced to come off the motorway on Friday night. A diversion using the A48 through Newport was put in place and the fire service tweeted that the M4 would stay closed until further notice while emergency repairs were carried out. Check if this is affecting your journey
options	0. wire      1. vehicle      2. link      3. valve      4. sign
<b>EC4</b>	<b>sample cases’ ambiguity</b>
question	A book about Adolf Hitler by a University of Aberdeen historian is to be turned into a @placeholder television series.
passage	Prof Thomas Weber’s book Hitler’s First War, which was released in 2010, claimed his image as a brave soldier was a myth. The producers of the Oscar-nominated film Downfall - also about the Nazi leader - will make the show after a French TV network purchased the series. The show will be called Hitler. Production of the 10-hour series begins next year. . .
options	0. major      1. thrilling      2. special      3. planned      4. forthcoming

Table 6: Examples from each fail case category. Options in green denotes gold answers, red denotes our system predictions. Passages are truncated to reserve the most relevant parts to the questions

## B Hyperparameter settings

Hyperparameter	Value
learning rate	1e-5
learning rate decay	linear
warmup fraction	0.1
Adam $\epsilon$	1e-6
Adam $\beta_1$	0.9
Adam $\beta_2$	0.999
gradient clip norm	1.0
Weight Decay	0.01
Dropout	0.1
Batch Size	32
Train Epochs	10 for task-adaptive pretraining, 5 for fine-tuning

Table 7: System Hyperparameter settings

# SemEval-2021 Task 5: Toxic Spans Detection

John Pavlopoulos<sup>†\*</sup>, Jeffrey Sorensen<sup>‡</sup>

Léo Laugier<sup>◇</sup>, Ion Androutsopoulos<sup>†</sup>

\* Department of Computer and System Sciences, Stockholm University, Sweden

† Department of Informatics, Athens University of Economic and Business, Greece

annis, ion@aueb.gr

◇ Télécom Paris, Institut Polytechnique de Paris, France

leo.laugier@telecom-paris.fr

‡ Google Jigsaw

sorenj@google.com

## Abstract

The Toxic Spans Detection task of SemEval-2021 required participants to predict the spans of toxic posts that were responsible for the toxic label of the posts. The task could be addressed as supervised sequence labeling, using training data with gold toxic spans provided by the organisers. It could also be treated as rationale extraction, using classifiers trained on potentially larger external datasets of posts manually annotated as toxic or not, without toxic span annotations. For the supervised sequence labeling approach and evaluation purposes, posts previously labeled as toxic were crowd-annotated for toxic spans. Participants submitted their predicted spans for a held-out test set, and were scored using character-based F1. This overview summarises the work of the 36 teams that provided system descriptions.

## 1 Introduction

Discussions online often host toxic posts, meaning posts that are rude, disrespectful, or unreasonable; and which can make users want to leave the conversation (Borkan et al., 2019a). Current toxicity detection systems classify whole posts as toxic or not (Schmidt and Wiegand, 2017; Pavlopoulos et al., 2017; Zampieri et al., 2019), often to assist human moderators, who may be required to review only posts classified as toxic, when reviewing all posts is infeasible. In such cases, human moderators could be assisted even more by automatically highlighting spans of the posts that made the system classify the posts as toxic. This would allow the moderators to more quickly identify objectionable parts of the posts, especially in long posts, and more easily approve or reject the decisions of the toxicity detection systems. As a first step along this direction, Task 5 of SemEval 2021 provided the participants with posts previously rated to be toxic, and required them to identify toxic spans,

i.e., spans that were responsible for the toxicity of the posts, when identifying such spans was possible. Note that a post may include no toxic span and still be marked as toxic. On the other hand, a non toxic post may comprise spans that are considered toxic in other toxic posts. We provided a dataset of English posts with gold annotations of toxic spans, and evaluated participating systems on a held-out test subset using character-based F1. The task could be addressed as supervised sequence labeling, training on the provided posts with gold toxic spans. It could also be treated as rationale extraction (Li et al., 2016; Ribeiro et al., 2016), using classifiers trained on larger external datasets of posts manually annotated as toxic or not, without toxic span annotations. There were almost 500 individual participants, and 36 out of the 92 teams that were formed submitted reports and results that we survey here. Most teams adopted the supervised sequence labeling approach. Hence, there is still scope for further work on the rationale extraction approach. We also discuss other possible improvements in the definition and data of the task.

## 2 Competition Dataset Creation

During 2015, when many publications were closing down comment sections due to moderation burdens, a start up named Civil Comments launched (Finley, 2016). Using a system of peer-based review and flagging, they hoped to crowd source the moderation responsibility. When this effort shut down in 2017 (Bogdanoff, 2017), they cited the financial constraints of the competitive publishing industry and the challenges of attaining the necessary scale.

The founders of Civil Comments, in collaboration with researchers from Google Jigsaw, undertook an effort to open source the collection of more than two million comments that had been collected. After filtering the comments to remove personally

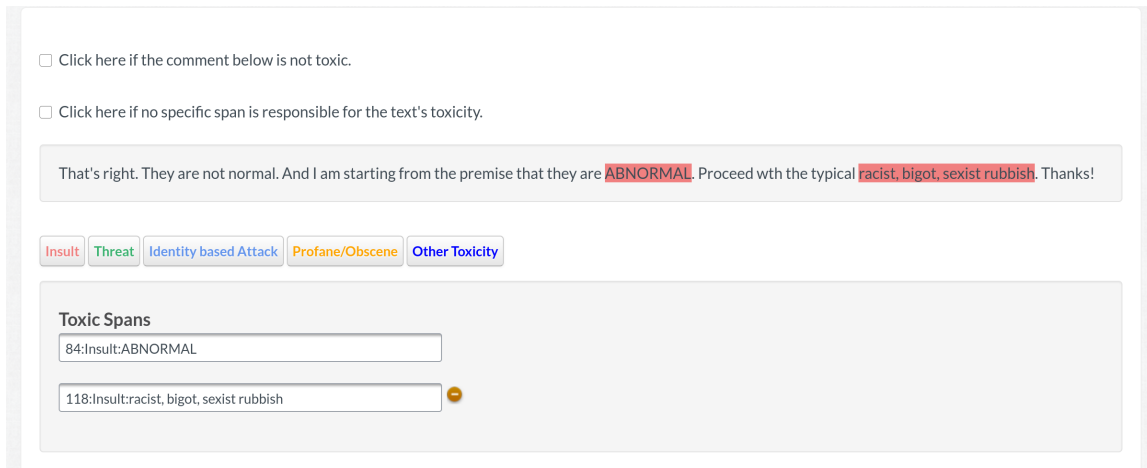


Figure 1: Screenshot of the Appen labeling interface that was used to annotate toxic spans.

identifiable information, a revised version of the annotation system of Wulczyn et al. (2017) was used on the Appen crowd rating platform to label the comments using a number of attributes including ‘toxicity’, ‘obscene’, ‘threat’ Borkan et al. (2019a). The complete dataset, partitioned into training, development, and test sets, was featured in a Kaggle competition,<sup>1</sup> with additional material, including individual rater decisions, published (Borkan et al., 2019b) after the close of the competition.

Civil Comments contains about 30k comments marked as toxic by a majority of at least three crowd raters. Toxic comments are *rare*, especially in fora that are not anonymous and where people have expectations that moderators will be watching and taking action. We undertook an effort to re-annotate this subset of comments at the span level, using the following instructions:

*For this task you will be viewing comments that a majority of annotators have already judged as toxic. We would like to know what parts of the comments are responsible for this.*

*Extract the toxic word sequences (spans) of the comment below, by highlighting each such span and then clicking the right button. If the comment is not toxic or if the whole comment should have been annotated, check the appropriate box and do not highlight any span.*

and a custom JavaScript based template,<sup>2</sup> which allowed selection and tagging of comment spans

<sup>1</sup>[www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification](http://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification)

<sup>2</sup>[github.com/ipavlopoulos/toxic\\_spans](https://github.com/ipavlopoulos/toxic_spans)

(Fig. 1). While raters were asked to categorize each span as one of five different categories, this was primarily intended as a priming exercise and all of the highlighted spans were collapsed into a single category. The lengths of the highlighted spans were decided by the raters. Seven raters were employed per post, but there were posts where fewer were eventually assigned. On the test subset (Table 1), we verified that the number of raters per post varied from three to seven; on the trial and train subsets this number varied from two to seven. All raters were warned the content might be explicit, and only raters who allowed adult content were selected.<sup>3</sup>

## 2.1 Inter-annotator Agreement

We measured inter-annotator agreement, initially, on a small set of 35 posts and we found 0.61 average Cohen’s Kappa. That is, we computed the mean pairwise Kappa per post, by using character offsets as instances being classified in two classes, toxic and non-toxic. And then we averaged Kappa over the 35 posts. On later experiments with larger samples (up to 1,000 posts) we observed equally moderate agreement and always higher than 0.55. Given the highly subjective nature of the task we consider this agreement to be reasonably high.

## 2.2 Extracting the ground truth

Each post comprises sets of annotated spans, one per rater. Each span is assigned a binary (toxic, non-toxic) label, based on whether the respective rater

<sup>3</sup>The full dataset and annotations for ToxicSpans is released ([github.com/ipavlopoulos/toxic\\_spans](https://github.com/ipavlopoulos/toxic_spans)) with a CC0 licence. The previously released Civil Comments dataset, on which the new dataset is based, was filtered to remove any potential personally identifiable information.



	<b>Trial</b>	<b>Train</b>	<b>Test</b>
<b>Number of posts</b>	690	7,939	2,000
<b>Avg. post length</b>	199.47	204.57	186.41
<b>Avg. toxic span length</b>	10.78	13.11	7.89
<b>Avg. # of toxic spans</b>	1.43	1.39	0.92

Table 1: Statistics of the trial, training, and test subsets of the dataset. Lengths are calculated in characters.

found the span to be insulting, threatening, identity-based attack, profane/obscene, or otherwise toxic. If the span was annotated with any of those types, the span is considered toxic according to the rater, otherwise not. For each post, we extracted the character offsets of each toxic span of each rater. In each post, the ground truth considers a character offset as toxic if the majority of the raters included it in their toxic spans, otherwise the ground truth of the character offset is non-toxic. A toxic span (Table 1) in the ground truth of a post is a maximal sequence of contiguous toxic character-offsets.

### 2.3 Exploratory analysis

After discarding duplicates and posts used as quiz questions to check the reliability of candidate annotators, we split the data into trial, train, and test (Table 1). Compared to the trial and training sets, the test set comprises posts with fewer characters and spans, but also shorter spans on average.

When studying the toxicity subtypes, we find that the vast majority of posts are annotated as insulting. In the training set, more than 6,000 posts are annotated as insulting, and the same high fraction is observed in the trial and test sets. Most of the toxic spans in the training set are single-word terms. The most frequent of them, such as ‘stupid’ and ‘idiot’, occur hundreds of times and remain frequent in the trial and test sets. Multi-word terms, such as ‘white trash’, ‘mentally ill’, are less frequent and vary across the three sets.

In an analysis of the test set, Palomino et al. (2021) used an emotion classifier that returns five scores per post, one for each of the following emotions: *anger*, *happiness*, *sadness*, *surprise*, *fear*.<sup>4</sup> *Fear* and *sadness* were reported to be the emotions with the highest average scores, a finding that we verified by repeating the experiment (see Fig. 2).<sup>5</sup> Interestingly, the emotion with the highest average score after *sadness* and *fear* is *surprise*, not *anger*, and *happiness* has the lowest score.

<sup>4</sup>[pypi.org/project/text2emotion](https://pypi.org/project/text2emotion)

<sup>5</sup>A post with a high *sadness* score (100%) is the following: “Such thin skin. **Pathetic.**”; the toxic span shown in red.

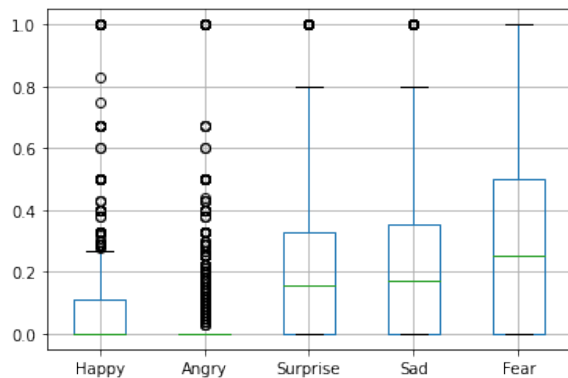


Figure 2: Emotion scores of the test posts. Emotion scores were obtained using an off-the-shelf emotion classifier, following Palomino et al. (2021).

## 3 Task description

The objective of this task is the detection of the spans that make a post toxic, when detecting such spans is possible. Systems had to extract a list of toxic spans, or an empty list, per post. A toxic span was defined to be a sequence of words that attribute to the post’s toxicity. Although we defined the task at the word level, gold labels were provided at the character level counting from zero (see Table 2).

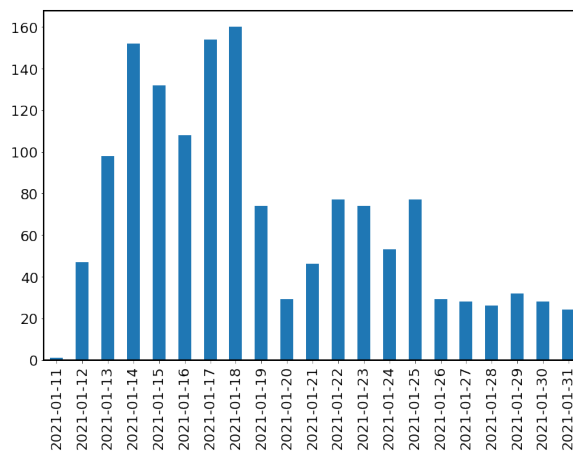


Figure 3: Number of submissions per evaluation day.

The evaluation period started on January 10, 2021 and finished on January 31, 2021. The first week, 10 submissions were allowed per day per team. The second week this number was reduced to 5, and it was reduced further to 1 during the final week. We chose to allow an extended evaluation period combined with multiple team submissions, to promote the competition. However, we also chose a decreasing submission limit to make it harder for participants to overfit the test set. As shown in Fig. 3, the number of submissions dropped over time, due to this constraint, but the interest was

OFFSETS	POST
{}	I will stand by my statement that liberalism is what smart looks like to stupid people. The letter posted yesterday states the real problem clearly enough. "It doesn't matter if it was protected by the First Amendment." I have legal news flash for the University of Oregon. Because the University is public and receives government tax dollars it is indeed bound to abide by the US constitution with ALL OF IT'S PROTECTION INCLUDED. Which should tell everyone right off the signatories are anti-American, and perhaps even anti-America in their twisted PC nonsense.
{}	Fits the despicable piece of trash like a glove.
{}	No he shot someone so he shouldn't be free. He should be locked up like He is for the next decade. You holla like its no big deal but what if it was your little brother who was shot? Then your dumb ass wouldnrht be screaming free ace g or whatever gay nickname he has. That is the lamest gangster name ever. Do you think that name scares people? Ace clown is what i will now call him.
{55, 56, 57, 58}	Time to give the opposition weapons and overthrow this <b>jerk</b> .
{31, ..., 39}	you are a paid liberal hack of <b>stupidity</b> shore living the liberal lie every day
{9, ..., 13}	Youre an <b>idiot</b> .
{7, 8, ..., 11, 14, 15, ..., 18}	what a <b>dummy, dummy</b> has no regard for the law. you cannot brutalize a suspect. he has complete lack of respect for any law and is acting like a dictator. he is trying to emulate putin.
{12, ..., 17, 94, ..., 102}	People make <b>stupid</b> decisions and then expect the gov't to bail them out. There is no cure for <b>stupidity</b> .
{14, ..., 20, 29, ..., 35}	Nah, the only <b>asshole</b> is the <b>asshole</b> firing a rifle within city limits.

Table 2: Examples of toxic test posts and their ground truth toxic spans (shown in red). The left column shows the character offsets of the toxic spans. The top three posts have no toxic spans, the next three have one each, while the remaining three posts have two toxic spans each.

continuous, and there were submissions until the last day. Despite the decreasing total number of submissions per day, the top daily score increased, reaching its maximum on the last day (see Fig. 4).

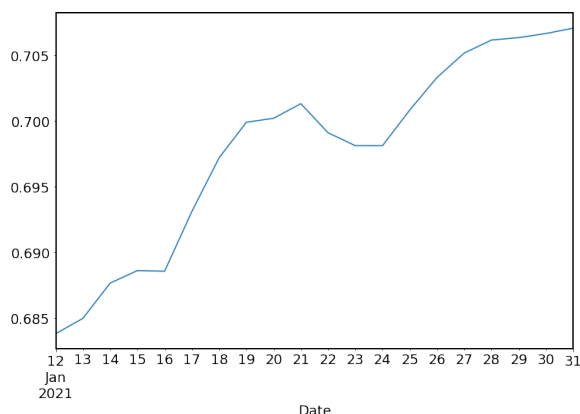


Figure 4: The evaluation score (character F1) of the best submission per day during the evaluation period.

## 4 Participation overview

We received 479 individual participation requests, 92 team formations, and 1,449 submissions. 91 teams submitted valid predictions (1,385 valid submissions in total) and were scored; out of these, only 36 submitted system descriptions.

### 4.1 The HITSZ-HLT submission

The best performing team (HITSZ-HLT) formulated the problem as a combination of token label-

ing and span extraction (Zhu et al., 2021).

For their token labeling approach, the team used two systems based on BERT (Devlin et al., 2019). Both systems had a Conditional Random Field (CRF) layer (Sutton and McCallum, 2006) on top, but one of the two also had an LSTM layer (Hochreiter and Schmidhuber, 1997) between BERT and the CRF layer. In both approaches, word-level BIO tags were used, i.e., words were labelled as B (beginning word of a toxic span), I (inside word of a toxic span), or O (outside of any toxic span).

For their span extraction approach, the team also used BERT. Roughly speaking, in this case BERT produces probabilities indicating how likely it is for each token to be the beginning or end of a toxic span. Then a heuristic search algorithm, originally developed for target extraction in sentiment analysis by Hu et al. (2019), selects the best combinations of candidate begin and end tokens, aiming to output the most likely set of toxic spans per post.

The character predictions of the three systems described above were combined with majority voting per character. That is, if any two systems considered a character to be part of a toxic span, then the ensemble classified the character as toxic, otherwise the ensemble classified it as non-toxic.

### 4.2 The S-NLP submission

The team with the second best performing system (S-NLP) consists of individual participants who grouped and submitted an ensemble of their sys-

tems (Nguyen et al., 2021). The ensemble combines two approaches, both of which are based on a RoBERTa model (Liu et al., 2019). The latter is first fine-tuned to classify posts as toxic or non-toxic, using three Kaggle toxicity datasets.<sup>6</sup> For toxic span detection, RoBERTa’s subword representations from three different layers (1, 6, 12) are summed to produce the corresponding word embeddings. A binary classifier on top of RoBERTa, operating on the word embeddings, predicts whether a word belongs to a toxic span or not.

For the first component of the ensemble, the word embeddings obtained from RoBERTa’s subword representations are concatenated with FLAIR (Akbik et al., 2019) and FastText (Bojanowski et al., 2017) embeddings.<sup>7</sup> The resulting embeddings are passed on to a two-layer stacked BiLSTM with a CRF layer on top to generate a BIO tag per word.

The second component of the ensemble used the RoBERTa model as a teacher to produce silver toxic spans for 30,000 unlabelled toxic posts (Borkan et al., 2019a). RoBERTa was then retrained as a student on the augmented dataset (30k posts with silver labels and the training posts provided by the organisers) to predict toxic offsets.

The ensemble returns the intersection of the toxic spans identified by the two components.

### 4.3 Additional interesting approaches

We now discuss some of the most interesting alternative approaches tried by the participants, even if they did not lead to high scores.

**Rationales** Some participants experimented with training toxicity classifiers on external datasets containing posts labeled as toxic or non-toxic; and then employing model-specific or model-agnostic rationale extraction mechanisms to produce toxic spans as explanations of the decisions of the classifier. The model-specific rationale mechanism of Rusert (2021) used the attention scores of an LSTM toxicity classifier to detect the toxic spans. Pluciński and Klimczak (2021) used the same approach, but also employed an orthogonalisation technique (Mohan Kumar et al., 2020). The model-agnostic rationale mechanism of Rusert (2021) combined an LSTM classifier with a token-masking approach that we call Input Erasure (IE), due to its similarities to the method of Li et al. (2016). The

<sup>6</sup>[github.com/unitaryai/detoxify](https://github.com/unitaryai/detoxify)

<sup>7</sup>In the latter case, in-vocabulary word embeddings were imported to Word2Vec for efficiency, and out of vocabulary words were handled with BPEs (Sennrich et al., 2016).

model-agnostic approach of Pluciński and Klimczak (2021) combined SHAP (Lundberg and Lee, 2017) with a fine-tuned BERT model. Ding and Jurgens (2021) and Benlahbib et al. (2021) also experimented with model-agnostic approaches, but they combined LIME (Ribeiro et al., 2016) with a Logistic Regression (LR) or with a linear Support Vector Machine (SVM) toxicity classifier. All the above mentioned approaches used a threshold to turn the explanation scores (e.g., attention or LIME scores) of the words into binary decisions (toxic/non-toxic words).

**Lexicon-based** No team relied on a purely lexicon-based approach, but few experimented with lexicon-based baselines (Zhu et al., 2021; Palomino et al., 2021) or used such components in ensembles (Ranasinghe et al., 2021). Three kinds of lexicon-based methods were used. First, the lexicon was handcrafted by domain experts (Smedt et al., 2020) and it was simply employed as a list of toxic words for lookup operations (Palomino et al., 2021). Second, the lexicon was compiled using the set of tokens labeled as toxic in our span-annotated training set and it was used as a lookup table (Burtenshaw and Kestemont, 2021), possibly also storing the frequency of each lexicon token in the training set (Zhu et al., 2021). The former two were also combined (Ranasinghe et al., 2021). Third, the least supervised lexicons were built with statistical analysis on the occurrences of tokens in a training set solely annotated at the comment level (toxic/non-toxic post) (Rusert, 2021). An added value of these approaches is that easy to use resources (toxicity lexicons) are built and shared publicly, such as the one suggested by Pluciński and Klimczak (2021).<sup>8</sup>

**Custom losses** Zhen Wang and Liu (2021) experimented with a new custom loss, which weighted false toxicity predictions based on their location in the text. If a false prediction was located near a ground truth toxic span, then it would contribute less to the overall loss for that post, compared to one located further away. The loss function used by Kuyumcu et al. (2021) to train their system is the Tversky Similarity Index (Tversky, 1977), a generalisation of the Sørensen–Dice coefficient and the Jaccard index, which was adjusted by the authors to weigh up false negatives.

**Data augmentation** The vast majority of the participating teams employed additional training data annotated at the post level. That is, either to

<sup>8</sup>[github.com/Orthrux-Lexicon/Toxic](https://github.com/Orthrux-Lexicon/Toxic)

build lexicons (Rusert, 2021), to leverage unsupervised rationale extraction methods (Rusert, 2021; Pluciński and Klimczak, 2021; Ding and Jurgens, 2021; Benlahbib et al., 2021), or to filter posts (Luu and Nguyen, 2021) that were not labeled as toxic by a toxicity classifier. Suman and Jain (2021) astutely produced silver data from external sources to augment the initial golden annotated dataset, training their model iteratively in a semi-supervised manner.

## 5 Evaluation

This section focuses on the evaluation framework of the task. First, the official measure that was used to evaluate the participating systems is described. Then, we discuss baseline models that were selected as benchmarks for comparison reasons. Finally, the results are presented.

### 5.1 Official evaluation measure

Following the work of Martino et al. (2019), systems were evaluated in terms of F1 computed on character offsets. For each system, we computed the F1 score per post, between the predicted and the ground truth character offsets. Then, we returned the macro-averaged (over test posts) score. When the ground truth set of character offsets was empty, we assigned a perfect score ( $F_1 = 1$ ) to the post in question if the predicted set of character offsets was also empty, and a zero score otherwise.<sup>9</sup>

### 5.2 Benchmarks

We report the results of some baselines, developed by us or the participants, to act as benchmarks.

BENCHMARK I was developed by Nguyen et al. (2021). It is based on a RoBERTa model, fine-tuned to predict if a post is toxic or not (Section 4.2) and further fine-tuned to predict toxic spans by using a CRF layer on top.

BENCHMARK II is a lexicon-based system, developed by Zhu et al. (2021), which extracts likely toxic words from the training data and simply tags them during inference. The lexicon comprises words that appear frequently inside ground truth toxic spans and not outside.

BENCHMARK III is a random baseline, which assigns a random label (toxic/non-toxic) per character offset (50% chance of being toxic).<sup>10</sup>

<sup>9</sup>The evaluation code can be found in our GitHub repository ([github.com/ipavlopoulos/toxic\\_spans](https://github.com/ipavlopoulos/toxic_spans)).

<sup>10</sup>The code of this baseline is also in the task’s repository.

## 5.3 Results

RANK	TEAM	SCORE (%)
1	HITSZ-HLT	70.83
2	S-NLP	70.77
<b>BASELINE</b>	<b>BENCHMARK I</b>	<b>69.89</b>
3	hitmi&t	69.85
5	YNU-HPCC	69.63
7	Cisco	69.22
8	MedAI	69.03
9	IITKDetox	68.95
13	GHOST	68.59
14	HLE-UPC	68.54
15	UTNLP	68.44
16	YoungSheldon	68.42
17	Lone Pine	68.38
18	sk	68.32
20	WLV-RIT	68.01
21	CSECUDSG	67.95
22	LISAC FSDM USMBA	67.84
23	UoT-UWF-PartAI	67.70
25	uob	67.61
<b>MEDIAN</b>	<b>The median score</b>	<b>67.58</b>
26	UAntwerp	67.55
27	MIPT-NSU-UTMN	67.55
28	NLRG	67.53
30	HamiltonDinggg	67.15
33	Iz1904	67.00
34	UIT-E10dot3	66.99
36	UniParma	66.72
37	hub	66.40
38	GoldenWindPlymouth	66.37
41	AStarTwice	66.16
44	sefamerve_arge	66.01
46	UPB	65.73
49	Entity	65.61
<b>BASELINE</b>	<b>BENCHMARK II</b>	<b>64.98</b>
57	BennettNLP (Fuchsia)	64.53
58	TeamGriek	64.31
63	UIT-ISE-NLP	62.23
75	NLP-Ulowa	50.09
<b>BASELINE</b>	<b>BENCHMARK III</b>	<b>12.22</b>
90	macech	7.33

Table 3: Official rank and F1 score (%) of the 36 participating teams that submitted system description papers. (There were 91 teams with submissions in total.) The median is shown in blue and benchmarks in red.

Table 3 shows the scores and ranks of all participating teams that described their approach, i.e., 36 out of 91 teams that participated. HITSZ-HLT (Section 4.1) was ranked first, followed by S-NLP (Section 4.2) that scored 0.06% lower. The rest of the teams followed with scores lower than 70%.

The score of the median is 67.58%, which is not far below the top scored team (-3.22 percent units), while it is far above the last two (+17.52 percent units). The standard deviation of system scores above the median is much lower (0.94) than that of the systems below the median (4.12). Most teams that were excluded from the table (because they did not describe their methods) score lower than

the median. However, there were also top scoring teams among those that were excluded, such as a team with a RoBERTa-based token-level ensemble that was ranked 4th.<sup>11</sup>

BENCHMARK I achieves a considerably high score and, hence, is very highly ranked. Combining BERT with a CRF or a span extraction method (two of the individual methods of the HITSZ-HLT ensemble, Section 4.1, not shown in Table 3) also performs well (Zhu et al., 2021), but these methods would be ranked two positions lower than BENCHMARK I. Nguyen et al. (2021) explored the benefits of further enhancing these word embeddings by concatenating them with FLAIR (Akbik et al., 2019) and FastText (Bojanowski et al., 2017) embeddings (Section 4.2). As shown in Fig. 5, the F1 score is slightly improved, reaching a maximum when both FLAIR and FastText embeddings are added.<sup>12</sup> We note that the same beneficial effect of enhancing the word embeddings was reported when using BERT as the base model (Sans and Farràs, 2021).

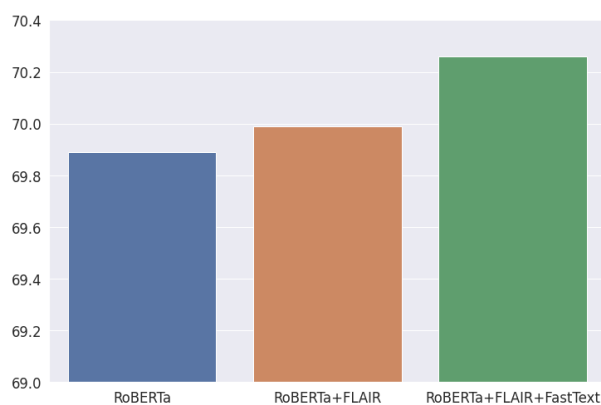


Figure 5: F1 of BENCHMARK I (Zhu et al., 2021) when FLAIR and FastText word embeddings are concatenated with the embeddings obtained from RoBERTa’s subword representations (from layers 1, 6, 12).

The lexicon-based BENCHMARK II and the random BENCHMARK III scored very low. The latter outperformed only one submission (MACECH), which sent the predictions in the wrong order. As noted in their report (Cech, 2021), if the predictions had been submitted in the correct order, the team’s score would have been 54%, and BENCHMARK III would have been the worst system in Table 3.

<sup>11</sup>We asked for details from participants that did not submit a description paper, but not all of them replied.

<sup>12</sup>Out of vocabulary words were tackled by using FastText embeddings of BPEs; consult Nguyen et al. (2021).

## 6 Analysis and discussion

Overall the organisers were happy to see the degree of involvement in this shared task, and the resulting diversity of approaches to this problem. We include some of our observations regarding the administration of the evaluation and what we have learned from the results.

### 6.1 Participation

The authors reached out to teams that decided not to submit a description paper and the vast majority were students who were time-limited. The fact that students participated in the task is promising and we plan to consider more ways to introduce SemEval tasks in classrooms. On the other hand, 60% of the participants chose not to describe their approach, which is problematic and should be addressed. A team could take advantage of such an option to create duplicate submissions and bypass any submission limits. More importantly, potentially interesting approaches are not discussed and properly compared to others.

It is also worth mentioning that the extended timeline allowed participants to join forces. For instance, a number of participants decided to combine their systems and form the 2nd ranked S-NLP. Their ensemble scored higher than all their standalone systems, though their best standalone system would still be ranked 2nd. In any case, we welcome the collaboration between participants, which may provide further insights regarding effective combinations of architectures.

### 6.2 General remarks on the approaches

Except for lexicon-based baselines, we observed that the vast majority of systems adopted the recent paradigm in NLP: fine-tuning large off-the-shelf Transformers (Vaswani et al., 2017) pre-trained on massive corpora. Non-Transformer based approaches, mostly LSTMs with pre-trained word embeddings were also used. The nature of the task, similar to the well-studied Named Entity Recognition (NER) task, led many competitors to use a CRF layer on top of the model (e.g., Transformers or LSTMs) of their choice.

### 6.3 Performance

The winning team (HITSZ-HLT) combined BERT with two approaches for their ensemble: a token labeling approach (two versions, with/without an LSTM between BERT and the CRF) and a span ex-

traction approach (Section 4.1). The comparison of the two showed that span extraction is slightly better on posts with a single span, but token labeling is clearly better on multi-span posts (Zhu et al., 2021). The complementary nature of the two approaches is probably what makes even a simple majority voting ensemble better than its competitors.

The system that was ranked second (S-NLP) also employed an ensemble, using a RoBERTa model initially fine-tuned to classify posts as toxic or non-toxic as the starting point (Nguyen et al., 2021). The ensemble combined (i) the resulting RoBERTa model, now fine-tuned to predict toxic spans, with additional FLAIR and FastText embeddings, and (ii) a RoBERTa model retrained as a student to predict toxic spans (Section 4.2). Although the two standalone models achieved higher scores than the standalone models of the top-ranked team (HITSZ-HLT), the ensemble did not yield significant improvements. This may be due to the student’s decisions not being that complementary to the teacher’s, as the team notes (Nguyen et al., 2021).

TBC	RE	F1 (%)	Report
LSTM	IE	38.29	Rusert (2021)
LSTM	ATT	49.70	Pluciński and Klimczak (2021)
LSTM	ATT	50.07	Rusert (2021)
LR	LIME	58.88	Benlahbib et al. (2021)
SVM	LIME	59.21	Benlahbib et al. (2021)
BERT	SHAP	<b>59.87</b>	Pluciński and Klimczak (2021)

Table 4: F1 on the evaluation set for systems employing rationale extraction (RE) mechanisms combined with post-level toxicity binary classifiers (TBC). Rationales are obtained via Input Erasure (IE), Attention (ATT), LIME, or SHAP. The binary classifier is an LSTM, Logistic Regression (LR), SVM, or BERT.

Teams that experimented with rationale extraction mechanisms (Section 4.3) did not find this approach advantageous compared to supervised sequence labeling in terms of F1 scores. However, the reported results of the rationale-based systems show that this approach is promising, especially because it does not require any data annotated at the span-level. Hence, there is scope for future work that could explore this direction further. Table 4 shows the F1 scores of all the rationale-based systems that were reported by participants. The binary toxic post classifiers that were used were LSTM, Logistic Regression (LR), Support Vector Machines (SVM), and BERT. The attention scores of an LSTM were used with (Pluciński and Klimczak, 2021) and without an orthogonality method (Rusert, 2021), with the latter being slightly bet-

ter; these are model-specific rationale extraction methods (Section 4.3). Model-agnostic approaches (Input Erasure, LIME, SHAP) were better than the model-specific ones. The best rationale-based method employed a BERT model, fine-tuned for toxic post classification, and SHAP.

Lexicon Name	F1 (%)	Report
WIEGAND 1 †	33.07	Zhu et al. (2021)
WORD-MATCH	40.86	Ranasinghe et al. (2021)
FREQ-RATIO †	41.55	Rusert (2021)
LOOKUP ‡	41.61	Burtenshaw and Kestemont (2021)
WIEGAND 2 †	50.98	Zhu et al. (2021)
ORTHRUS	61.07	Palomino et al. (2021)
HITSZ-HLT ‡	<b>64.98</b>	Zhu et al. (2021)
+WORDNET	64.09	Zhu et al. (2021)
+GLOVE	64.19	Zhu et al. (2021)

Table 5: F1 on the evaluation set for lexicon-based systems. Systems that are followed by † and ‡ use exclusively external and internal resources respectively.

Lexicon-based approaches were only used as baselines or components in ensembles, as already noted. In principle, all lexicon-based systems are extremely efficient and interpretable. Table 5 shows they can also achieve surprisingly high scores. Recall that we used the best performing lexicon-based system, developed by Zhu et al. (2021), as BENCHMARK II. Its score is included in Table 3. Despite the fact that it is low ranked, its F1 score is less than 6 percent points lower than that of the best submission. We also note that BENCHMARK II is a high-precision classifier; it outperforms even the best system in terms of precision (Zhu et al., 2021). Attempts to expand its lexicon using WordNet and GloVe, improved recall, but eventually harmed precision and its F1 score.

## 6.4 Error analysis

A common theme across many competitor reports was the serious challenge posed by comments with no toxic spans. It is not readily evident why this is a common occurrence in the task, and certainly the way that annotation consensus is used to combine annotations can be a contributing factor. However, many systems seemed determined to tag *some* spans and many authors noted that performance on posts with no tagged span was extremely poor compared to performance on posts with tagged spans.

Many systems were also reluctant to tag function words like ‘of’ and ‘and’, which can be included in multi-word spans (e.g., ‘piece of crap’), leading to a decline in performance as measured by the chosen F1 measure. The overwhelming presence

of single word gold spans in the training set favors short spans. But the majority of the short spans comprises common cuss or clearly abusive words, which can be directly classified as toxic (Ghosh and Kumar, 2021); by contrast, the infrequent longer spans are rather context dependent and more challenging to detect. This probably also contributed to the performance of the best system (HITSZ-HLT), since one of the two components of that ensemble handled better long spans, as already discussed in Section 6.3.

Other error analysis highlighted challenges intrinsic to the task. The strong dependency of toxicity on context makes it particularly difficult to solve with systems based on vocabulary. Toxicity, when expressed with subtle language, can appear through non-local text features: some comments are toxic without showing any obvious toxic span in them. Such posts made the task more difficult for participants, because systems had learnt to label the words bearing the most negative sentiment (Bansal et al., 2021). Annotation mistakes were also reported (Table 6).

Type	Description
INCONSISTENCIES	Not all the occurrences of the same toxic span are annotated in the same post.
FALSE NEGATIVES	Toxic words missed.
FALSE POSITIVES	Non-toxic words labelled.

Table 6: The types and descriptions of the annotation mistakes that were detected by some of the participants.

Participants that were notable for their effort in error analysis include Bansal et al. (2021), Hoang and Nguyen (2021), Ding and Jurgens (2021), and Ghosh and Kumar (2021), where an additional effort was made to examine their model’s ability to correctly tag words in toxic and non-toxic contexts. Interestingly Sans and Farràs (2021) also noted in their analysis that racial and ethnic terms are labeled in biased ways that reflect patterns not only in the training toxic spans, but also in external data used to pre-train underlying Transformer models.

## 7 Conclusions

We provided 10,629 posts that were annotated for toxic spans and we defined the task of toxic span detection. The task was popular, attracting almost 500 individual participants. Eventually 91 teams were formed, out of which 36 submitted a description report. This overview described the approaches of these 36 teams and discussed their results.

Pre-trained Transformers, fine-tuned by viewing the task as a sequence labelling one, performed well and solutions that combined these models within an ensemble were highly-rated. The performance of these models increases further with the help of pre-trained word embeddings or by using multiple Transformer layers to embed words.

Long toxic spans were more likely context-dependent and less frequent in the dataset compared to single-word spans, which made their detection a challenge. The winners included in their ensemble an approach that performed better on long spans, but we note that the problem of detecting long uncommon toxic spans is far from solved.

Of particular interest were approaches that employed rationale extraction mechanisms, which do not require any training data annotated at the span level. They performed much worse than sequence labeling approaches, but this is a promising direction that was considered by only a few participants.

Future similar competitions could benefit from tracks that separate supervised from unsupervised solutions. The development of datasets created with the help of crowd annotators should focus on addressing ambiguity, bias, inconsistencies, and misannotations. This could be accomplished by adding more annotators per post. Future competitions could also require participants to both classify posts as toxic or not, and detect toxic spans only when posts are classified as toxic, instead of providing the participants only with posts already classified as toxic. Finally, future competitions could require participants to distinguish toxic posts of different kinds (e.g., insult, threat, profanity, along with supporting spans), which are sometimes easier to define compared to the more general umbrella toxicity term we (and others) have used.

## Acknowledgement

We thank the participants and the reviewers for their useful comments and suggestions. This research was funded in part by a Google Research Award.

## References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL Demonstrations*, pages 54–59.
- Archit Bansal, Abhay Kaushik, and Ashutosh Modi. 2021. IITK@Detox at SemEval-2021 Task 5: Semi-

- supervised learning and dice loss for toxic spans detection. In *SemEval*.
- Abdessamad Benlahbib, Hamza Alami, and Ahmed Alami. 2021. LISAC FSDM USMBA at SemEval 2021 Task 5: Tackling toxic spans detection challenge with supervised spanBERT-based model and unsupervised LIME-based model. In *SemEval*.
- Aja Bogdanoff. 2017. [Saying goodbye to civil comments](#). Accessed: 2021-04-15.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019a. Nuanced metrics for measuring unintended bias with real data for text classification. In *WWW*, pages 491–500, San Francisco, USA.
- Daniel Borkan, Jeffrey Sorensen, and Lucy Vasserman. 2019b. [Exploring the role of human raters in creating nlp datasets](#). Accessed: 2021-04-15.
- Ben Burtenshaw and Mike Kestemont. 2021. UAntwerp at SemEval-2021 Task 5: Spans are spans, stacking a binary word level approach to toxic span detection. In *SemEval*.
- Maggie Cech. 2021. macech at SemEval-2021 Task 5: Toxic spans detection. In *SemEval*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.
- Huiyang Ding and David Jurgens. 2021. HamiltonD-inggg at SemEval-2021 Task 5: Investigating toxic span detection using RoBERTa pre-training. In *SemEval*.
- Kline Finley. 2016. [Want to save the comments from trolls? do it yourself](#). Accessed: 2021-04-15.
- Sreyan Ghosh and Sonal Kumar. 2021. Cisco at SemEval-2021 Task 5: What’s toxic?: Leveraging transformers for multiple toxic span extraction from online comments. In *SemEval*.
- Phu Gia Hoang and Luan Thanh Nguyen. 2021. UIT-E10dot3 at SemEval 2021 Task 5: Toxic spans detection with roberta and spacy’s library base systems. In *SemEval*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. In *ACL*, pages 537–546.
- Birol Kuyumcu, Selman Delil, and Cüneyt aksakalli. 2021. Sefamerve\_arge at SemEval-2021 Task 5: Toxic span detection using segmentation based 1-d convolutional neural network model. In *SemEval*.
- Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.
- Son T. Luu and Ngan Nguyen. 2021. UIT-ISE-NLP at SemEval-2021 Task 5: Toxic span detection with BiLSTM - CRF and toxic BERT comment classification. In *SemEval*.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *EMNLP-IJCNLP*, pages 5640–5650.
- Akash Kumar Mohankumar, Preksha Nema, Sharan Narasimhan, Mitesh M Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2020. Towards transparent and explainable attention models. In *ACL*, pages 4206–4216.
- Viet Anh Nguyen, Tam Nguyen, Huy Dao Quang, and Quang Huu Pham. 2021. S-NLP at semeval-2021 task 5: Toxic spans detection. In *SemEval*.
- Marco Palomino, Dawid Grad, and James Bedwell. 2021. An ensemble approach to identify toxicity in text. In *SemEval*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *EMNLP*, pages 1125–1135, Copenhagen, Denmark.
- Kamil Pluciński and Hanna Klimczak. 2021. GHOST at SemEval-2021 Task 5: Is explanation all you need? In *SemEval*.
- Tharindu Ranasinghe, Diptanu Sarkar, Marcos Zampieri, and Alexander Ororbia. 2021. WLV-RIT at SemEval-2021 Task 5: A neural transformer framework for detecting toxic spans. In *SemEval*.
- Marco T. Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “Why Should I Trust You?” Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144, San Francisco, USA.
- Jonathan Rusert. 2021. NLP\_UIOWA at Semeval-2021 Task 5: Transferring toxic sets to tag toxic spans. In *SemEval*.



- Rafel Palliser Sans and Albert Rial Farràs. 2021. HLE-UPC at SemEval-2021 Task 5: Multi-Depth DistilBERT for toxic spans detection. In *SemEval*.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.
- Tom De Smedt, Pierre Voué, Sylvia Jaki, Melina Röttcher, and Guy De Pauw. 2020. Profanity & offensive words (POW). *Textgain*.
- Thakur Ashutosh Suman and Abhinav Jain. 2021. AS-tarTwice at SemEval-2021 Task 5: Toxic span detection using RoBERTa-CRF, domain specific pre-training and self-training. In *SemEval*.
- Charles Sutton and Andrew McCallum. 2006. An Introduction to Conditional Random Fields for relational learning. *Introduction to statistical relational learning*, 2:93–128.
- Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, volume 30.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *WWW*, pages 1391–1399, Perth, Australia.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *SemEval*.
- Hongjie Fan Zhen Wang and Junfei Liu. 2021. MedAI at SemEval-2021 Task 5: Start-to-end tagging framework for toxic spans detection. In *SemEval*.
- Qinglin Zhu, Zijie Lin, Yice Zhang, Jingyi Sun, Xiang Li, Qihui Lin, and Ruifeng Xu. 2021. HITSZ-HLT at SemEval-2021 Task 5: Span-based ensemble model with toxic lexicon. In *SemEval*.

# SemEval-2021 Task 6: Detection of Persuasion Techniques in Texts and Images

Dimitar Dimitrov,<sup>1</sup> Bishr Bin Ali,<sup>2</sup> Shaden Shaar,<sup>3</sup> Firoj Alam,<sup>3</sup>  
Fabrizio Silvestri,<sup>4</sup> Hamed Firooz,<sup>5</sup> Preslav Nakov,<sup>3</sup> and Giovanni Da San Martino<sup>6</sup>

<sup>1</sup> Sofia University “St. Kliment Ohridski”, Bulgaria, <sup>2</sup> King’s College London, UK,

<sup>3</sup> Qatar Computing Research Institute, HBKU, Qatar

<sup>4</sup> Sapienza University of Rome, Italy, <sup>5</sup> Facebook AI, USA, <sup>6</sup> University of Padova, Italy

mitko.bg.ss@gmail.com, bishrkc@gmail.com

{sshaar, fialam, pnakov}@hbku.edu.qa, mhfirooz@fb.com

fsilvestri@diag.uniroma1.it, dasan@math.unipd.it

## Abstract

We describe *SemEval-2021 task 6 on Detection of Persuasion Techniques in Texts and Images*: the data, the annotation guidelines, the evaluation setup, the results, and the participating systems. The task focused on memes and had three subtasks: (i) detecting the techniques in the text, (ii) detecting the text spans where the techniques are used, and (iii) detecting techniques in the entire meme, i.e., both in the text and in the image. It was a popular task, attracting 71 registrations, and 22 teams that eventually made an official submission on the test set. The evaluation results for the third subtask confirmed the importance of both modalities, the text and the image. Moreover, some teams reported benefits when not just combining the two modalities, e.g., by using early or late fusion, but rather modeling the interaction between them in a joint model.

## 1 Introduction

Internet and social media have amplified the impact of disinformation campaigns. Traditionally a monopoly of states and large organizations, now such campaigns have become within the reach of even small organisations and individuals (Da San Martino et al., 2020b).

Such propaganda campaigns are often carried out using posts spread on social media, with the aim to reach very large audience. While the rhetorical and the psychological devices that constitute the basic building blocks of persuasive messages have been thoroughly studied (Miller, 1939; Weston, 2008; Torok, 2015), only few isolated efforts have been made to devise automatic systems to detect them (Habernal et al., 2018; Habernal et al., 2018; Da San Martino et al., 2019b).

WARNING: This paper contains meme examples and wording that might be offensive to some readers.



Figure 1: A meme with a civil war threat during the President Trump’s impeachment trial. Two persuasion techniques are used: (i) *Appeal to Fear* in the image, and (ii) *Exaggeration* in the text. **Source(s):** Image ; License

Thus, in 2020, we proposed *SemEval-2020 task 11 on Detection of Persuasion Techniques in News Articles*, with the aim to help bridge this gap (Da San Martino et al., 2020a). The task focused on text only. Yet, some of the most influential posts in social media use memes, as shown in Figure 1,<sup>1</sup> where visual cues are being used, along with text, as a persuasive vehicle to spread disinformation (Shu et al., 2017). During the 2016 US Presidential campaign, malicious users in social media (bots, cyborgs, trolls) used such memes to provoke emotional responses (Guo et al., 2020).

In 2021, we introduced a new SemEval shared task, for which we prepared a multimodal corpus of memes annotated with an extended set of techniques, compared to SemEval-2020 task 11. This time, we annotated both the text of the memes, highlighting the spans in which each technique has been used, as well as the techniques appearing in the visual content of the memes.

<sup>1</sup>In order to avoid potential copyright issues, all memes we show in this paper are our own recreation of existing memes, using images with clear copyright.

Based on our annotations, we offered the following three subtasks:

**Subtask 1 (ST1)** Given the textual content of a meme, identify which techniques (out of 20 possible ones) are used in it. This is a multilabel classification problem.

**Subtask 2 (ST2)** Given the textual content of a meme, identify which techniques (out of 20 possible ones) are used in it together with the span(s) of text covered by each technique. This is a multilabel sequence tagging task.

**Subtask 3 (ST3)** Given a meme, identify which techniques (out of 22 possible ones) are used in the meme, considering both the text and the image. This is a multilabel classification problem.

A total of 71 teams registered for the task, 22 of them made an official submission on the test set and 15 of the participating teams submitted a system description paper.

## 2 Related Work

**Propaganda Detection** Previous work on propaganda detection has focused on analyzing textual content (Barrón-Cedeno et al., 2019; Da San Martino et al., 2019b; Rashkin et al., 2017). See (Martino et al., 2020) for a recent survey on computational propaganda detection. Rashkin et al. (2017) developed the TSHP-17 corpus, which had document-level annotations with four classes: *trusted*, *satire*, *hoax*, and *propaganda*. Note that TSHP-17 was labeled using distant supervision, i.e., all articles from a given news outlet were assigned the label of that news outlet. The news articles were collected from the English Gigaword corpus (which covers reliable news sources), as well as from seven unreliable news sources, including two propagandistic ones. They trained a model using word  $n$ -grams, and reported that it performed well only on articles from sources that the system was trained on, and that the performance degraded quite substantially when evaluated on articles from unseen news sources. Barrón-Cedeno et al. (2019) developed a corpus QProp with two labels (propaganda vs. non-propaganda), and experimented with two corpora: TSHP-17 and QProp. They binarized the labels of TSHP-17 as follows: propaganda vs. the other three categories.

They performed massive experiments, investigated writing style and readability level, and trained models using logistic regression and SVMs. Their findings confirmed that using distant supervision, in conjunction with rich representations, might encourage the model to predict the source of the article, rather than to discriminate propaganda from non-propaganda. The study by Habernal et al. (2017, 2018) also proposed a corpus with 1.3k arguments annotated with five fallacies, including *ad hominem*, *red herring*, and *irrelevant authority*, which directly relate to propaganda techniques.

A more fine-grained propaganda analysis was done by Da San Martino et al. (2019b), who developed a corpus of news articles annotated with the spans of use of 18 propaganda techniques, from an inventory they put together. They targeted two tasks: (i) binary classification —given a sentence, predict whether any of the techniques was used in it; and (ii) multi-label multi-class classification and span detection task —given a raw text, identify both the specific text fragments where a propaganda technique is being used as well as the type of technique. They further proposed a multi-granular gated deep neural network that captures signals from the sentence-level task to improve the performance of the fragment-level classifier and vice versa. Subsequently, an automatic system, Prta, was developed and made publicly available (Da San Martino et al., 2020c), which performs fine-grained propaganda analysis of text using these 18 fine-grained propaganda techniques.

**Multimodal Content** Another line of related research is on analyzing multimodal content, e.g., for predicting misleading information (Volkova et al., 2019), for detecting deception (Glenski et al., 2019), emotions and propaganda (Abd Kadir et al., 2016), hateful memes (Kiela et al., 2020), and propaganda in images (Seo, 2014). Volkova et al. (2019) developed a corpus of 500K Twitter posts consisting of images and labeled with six classes: disinformation, propaganda, hoaxes, conspiracies, clickbait, and satire. Glenski et al. (2019) explored multilingual multimodal content for deception detection. Multimodal hateful memes were the target of the *Hateful Memes Challenge*, which was addressed by fine-tuning state-of-art methods such as ViLBERT (Lu et al., 2019), Multimodal BERT (Kiela et al., 2019), and VisualBERT (Li et al., 2019) to classify hateful vs. not-hateful memes (Kiela et al., 2020).

**Related Shared Tasks** The present shared task is closely related to *SemEval-2020 task 11 on Detection of Persuasion Techniques in News Articles* (Da San Martino et al., 2020a), which focused on news articles, and asked (i) to detect the spans where propaganda techniques are used, as well as (ii) to predict which propaganda technique (from an inventory of 14 techniques) is used in a given text span. Another closely related shared task is the *NLP4IF-2019 task on Fine-Grained Propaganda Detection*, which asked to detect the spans of use in news articles of each of 18 propaganda techniques (Da San Martino et al., 2019a). While these tasks focused on the text of news articles, here we target memes and multimodality, and we further use an extended inventory of 22 propaganda techniques.

Other related shared tasks include the FEVER 2018 and 2019 tasks on *Fact Extraction and Verification* (Thorne et al., 2018), the SemEval 2017 and 2019 tasks on predicting the veracity of rumors in Twitter (Derczynski et al., 2017; Gorrell et al., 2019), the SemEval-2019 task on *Fact-Checking in Community Question Answering Forums* (Mihaylova et al., 2019), the NLP4IF-2021 shared task on *Fighting the COVID-19 Infodemic* (Shaar et al., 2021). We should also mention the CLEF 2018–2021 *CheckThat!* lab (Nakov et al., 2018; Elsayed et al., 2019a,b; Barrón-Cedeño et al., 2020; Barrón-Cedeño et al., 2020), which featured tasks on automatic identification (Atanasova et al., 2018, 2019) and verification (Barrón-Cedeño et al., 2018; Hasanain et al., 2019, 2020; Shaar et al., 2020; Nakov et al., 2021) of claims in political debates and social media. While these tasks focused on factuality, check-worthiness, and stance detection, here we target propaganda; moreover, we focus on memes and on multimodality rather than on analyzing the text of tweets, political debates, or community question answering forums.

### 3 Persuasion Techniques

Scholars have proposed a number of inventories of persuasion techniques of various sizes (Miller, 1939; Torok, 2015; Abd Kadir and Sauffiyan, 2014). Here, we use an inventory of 22 techniques, borrowing from the lists of techniques described in (Da San Martino et al., 2019b), (Shah, 2005) and (Abd Kadir and Sauffiyan, 2014). Among these 22 techniques, the first 20 are applicable to both text and images, while the last two, *Appeal to (Strong) Emotions* and *Transfer*, are reserved for images.

Below, we provide a definition for each of these 22 techniques; more detailed instructions of the annotation process and examples are provided in Appendix A.

1. **Loaded Language:** Using specific words and phrases with strong emotional implications (either positive or negative) to influence an audience.
2. **Name Calling or Labeling:** Labeling the object of the propaganda campaign as either something the target audience fears, hates, finds undesirable, or loves, praises.
3. **Doubt:** Questioning the credibility of someone or something.
4. **Exaggeration or Minimisation:** Either representing something in an excessive manner, e.g., making things larger, better, worse (“*the best of the best*”, “*quality guaranteed*”), or making something seem less important or smaller than it really is, e.g., saying that an insult was just a joke.
5. **Appeal to Fear or Prejudices:** Seeking to build support for an idea by instilling anxiety and/or panic in the population towards an alternative. In some cases, the support is built based on preconceived judgments.
6. **Slogans:** A brief and striking phrase that may include labeling and stereotyping. Slogans tend to act as emotional appeals.
7. **Whataboutism:** A technique that attempts to discredit an opponent’s position by charging them with hypocrisy without directly disproving their argument.
8. **Flag-Waving:** Playing on strong national feeling (or positive feelings toward any group, e.g., based on race, gender, political preference) to justify or promote an action or idea.
9. **Misrepresentation of Someone’s Position (Straw Man):** When an opponent’s proposition is substituted with a similar one, which is then refuted in place of the original proposition.
10. **Causal Oversimplification:** Assuming a single cause or reason, when there are actually multiple causes for an issue. It includes transferring blame to one person or group of people without investigating the actual complexities of the issue.

11. **Appeal to Authority:** Stating that a claim is true because a valid authority or expert on the issue said so, without any other supporting evidence offered. We consider the special case in which the reference is not an authority or an expert as part of this technique, although it is referred to as *Testimonial* in the literature.
12. **Thought-Terminating Cliché:** Words or phrases that discourage critical thought and meaningful discussion about a given topic. They are typically short, generic sentences that offer seemingly simple answers to complex questions or that distract the attention away from other lines of thought.
13. **Black-and-White Fallacy or Dictatorship:** Presenting two alternative options as the only possibilities, when in fact more possibilities exist. As an extreme case, tell the audience exactly what actions to take, eliminating any other possible choices (*Dictatorship*).
14. **Reductio ad Hitlerum:** Persuading an audience to disapprove of an action or an idea by suggesting that the idea is popular with groups that are hated or in contempt by the target audience. It can refer to any person or concept with a negative connotation.
15. **Repetition:** Repeating the same message over and over again, so that the audience will eventually accept it.
16. **Obfuscation, Intentional Vagueness, Confusion:** Using words that are deliberately not clear, so that the audience can have their own interpretations.
17. **Presenting Irrelevant Data (Red Herring):** Introducing irrelevant material to the issue being discussed, so that everyone’s attention is diverted away from the points made.
18. **Bandwagon** Attempting to persuade the target audience to join in and take the course of action because “everyone else is taking the same action.”
19. **Smears:** A smear is an effort to damage or call into question someone’s reputation, by propounding negative propaganda. It can be applied to individuals or groups.
20. **Glittering Generalities (Virtue):** These are words or symbols in the value system of the target audience that produce a positive image when attached to a person or an issue.
21. **Appeal to (Strong) Emotions:** Using images with strong positive/negative emotional implications to influence an audience.
22. **Transfer:** Also known as *Association*, this is a technique that evokes an emotional response by projecting positive or negative qualities (praise or blame) of a person, entity, object, or value onto another one in order to make the latter more acceptable or to discredit it.

## 4 Dataset

The annotation process is explained in detail in Appendix A, and in this section, we give a just brief summary.

We collected English memes from our personal Facebook accounts over several months in 2020 by following 26 public Facebook groups, which focus on politics, vaccines, COVID-19, and gender equality. We considered a meme to be a “*photograph style image with a short text on top of it*”, and we removed examples that did not fit this definition, e.g., cartoon-style memes, memes whose textual content was strongly dominant or non-existent, memes with a single-color background image, etc. Then, we annotated the memes using our 22 persuasion techniques. For each meme, we first annotated its textual content, and then the entire meme. We performed each of these two annotations in two phases: in the first phase, the annotators independently annotated the memes; afterwards, all annotators met together with a consolidator to discuss and to select the final gold label(s).

The final annotated dataset consists of 950 memes: 687 memes for training, 63 for development, and 200 for testing. While the maximum number of sentences in a meme is 13, the average number of sentences per meme is just 1.68, as most memes contain very little text.

Table 1 shows the number of instances of each technique for each of the tasks. Note that *Transfer* and *Appeal to (Strong) Emotions* are not applicable to text, i.e., to Subtasks 1 and 2. For Subtasks 1 and 3, each technique can be present at most once per example, while in Subtask 2, a technique could appear multiple times in the same example. This explains the sizeable differences in the number of instances for some persuasion techniques between Subtasks 1 and 2: some techniques are over-used in memes, with the aim of making the message more persuasive, and thus they contribute higher counts to Subtask 2.

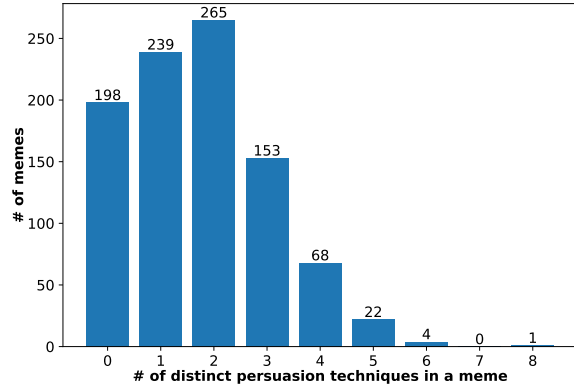
Persuasion Techniques	Subtask 1	Subtask 2	Subtask 3	
	#	Len.	#	#
Loaded Language	489	2.41	761	492
Name Calling/Labeling	300	2.62	408	347
Smears	263	17.11	266	602
Doubt	84	13.71	86	111
Exaggeration/Minimisation	78	6.69	85	100
Slogans	66	4.70	72	70
Appeal to Fear/Prejudice	57	10.12	60	91
Whataboutism	54	22.83	54	67
Glittering Generalities (Virtue)	44	14.07	45	112
Flag-Waving	38	5.18	44	55
Repetition	12	1.95	42	14
Causal Oversimplification	31	14.48	33	36
Thought-Terminating Cliché	27	4.07	28	27
Black-and-White	25	11.92	25	26
Fallacy/Dictatorship	24	15.96	24	40
Appeal to Authority	22	20.05	22	35
Reductio ad Hitlerum	13	12.69	13	23
Obfuscation, Intentional Vagueness, Confusion	5	9.8	5	7
Presenting Irrelevant Data	5	15.4	5	7
Bandwagon	5	8.4	5	5
Transfer	—	—	—	95
Appeal to (Strong) Emotions	—	—	—	90
<b>Total</b>	<b>1,642</b>	<b>2,119</b>	<b>2,488</b>	

Table 1: Statistics about the persuasion techniques. For each technique, we show the average length of its spans (in number of words) and the number of its instances as annotated in the text only vs. in the entire meme.

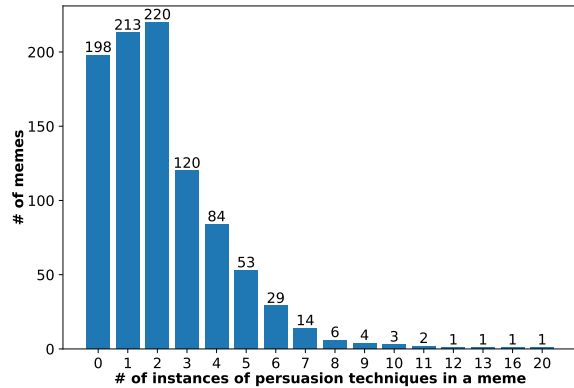
Note that the number of instances for Subtasks 1 and 3 differs, and in some cases by quite a bit, e.g., for *Smears*, *Doubt*, and *Appeal to Fear/Prejudice*. This shows that many techniques cannot be found in the text, and require the visual content, which motivates the need for multimodal approaches for Subtask 3. Note also that different techniques have different span lengths, e.g., *Loaded Language* and *Name Calling* are about 2–3 words long, e.g., *violence*, *mass shooter*, and *coward*. However, for techniques such as *Whataboutism*, the average span length is 22 words.

Figure 2 shows statistics about the distribution of the number of persuasion techniques per meme. Note the difference for memes without persuasion techniques between Figures 2a and 2c: we can see that the number of memes without any persuasion technique drastically drops for Subtask 3. This is because the visual modality introduces additional context that was not available during the text-only annotation, which further supports the need for multimodal analysis. The visual modality also has an impact on memes that already had persuasion techniques in the text-only phase.

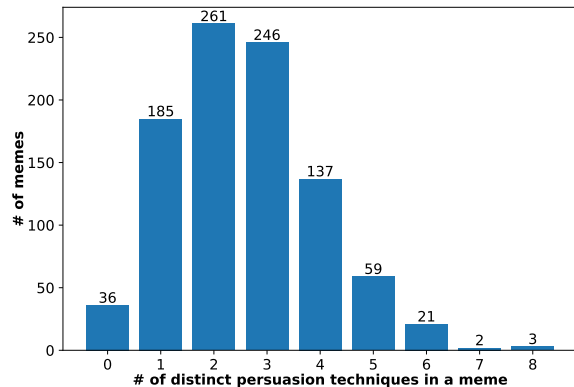
We observe that the number of memes with only one persuasion technique in Subtask 3 is considerably lower compared to Subtask 1, while the number of memes with three or more persuasion techniques has greatly increased for Subtask 3.



(a) Subtask 1



(b) Subtask 2



(c) Subtask 3

Figure 2: Distribution of the number of persuasion techniques per meme. Subfigure (b) reports the **number of instances** of persuasion techniques for a meme. Note that a meme could have multiple instances of the same technique for this subtask. Subfigures (a) and (c) show the number of **distinct** persuasion techniques in a meme.

## 5 Evaluation Framework

### 5.1 Evaluation Measures

**Subtasks 1 and 3** To measure the performance of the systems, for Subtasks 1 and 3, we use Micro and Macro  $F_1$ , as these are multi-class multi-label tasks, where the labels are imbalanced. The official measure for the task is Micro  $F_1$ .

**Subtask 2** For Subtask 2, the evaluation requires matching the text spans. Hence, we use an evaluation function that gives credit to partial matches between gold and predicted spans.

Let document  $d$  be represented as a sequence of characters. The  $i$ -th propagandistic text fragment is then represented as a sequence of contiguous characters  $t \subseteq d$ . A document includes a set of (possibly overlapping) fragments  $T$ . Similarly, a learning algorithm produces a set  $S$  with fragments  $s \subseteq d$ , predicted on  $d$ . A labeling function  $l(x) \in \{1, \dots, 20\}$  associates  $t \in T$ ,  $s \in S$  with one of the techniques. An example of (gold) annotation is shown in Figure 3, where an annotation  $t_1$  marks the span *stupid and petty* with the technique *Loaded Language*.

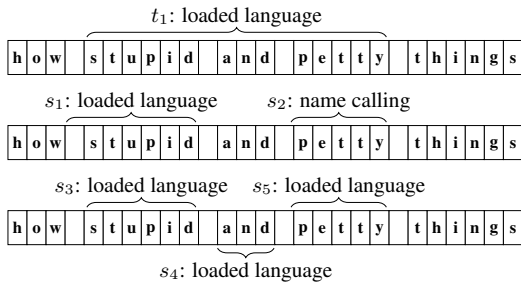


Figure 3: Example of gold annotation (top) and the predictions of a supervised model (bottom) in a document represented as a sequence of characters.

We define the following function to handle partial overlaps of fragments with the same labels:

$$C(s, t, h) = \frac{|(s \cap t)|}{h} \delta(l(s), l(t)), \quad (1)$$

where  $h$  is a normalizing factor and  $\delta(a, b) = 1$  if  $a = b$ , and 0, otherwise. For example, still with reference to Figure 3,  $C(t_1, s_1, |t_1|) = \frac{6}{16}$  and  $C(t_1, s_2, |t_1|) = 0$ .

Given Eq. (1), we now define variants of precision and recall that can account for the imbalance in the corpus:

$$P(S, T) = \frac{1}{|S|} \sum_{\substack{s \in S, \\ t \in T}} C(s, t, |s|), \quad (2)$$

$$R(S, T) = \frac{1}{|T|} \sum_{\substack{s \in S, \\ t \in T}} C(s, t, |t|), \quad (3)$$

We define (2) to be zero if  $|S| = 0$ , and Eq. (3) to be zero if  $|T| = 0$ . Following Potthast et al. (2010), in (2) and (3) we penalize systems predicting too many or too few instances by dividing by  $|S|$  and  $|T|$ , respectively. Finally, we combine Eqs. (2) and (3) into an  $F_1$ -measure, the harmonic mean of precision and recall.

### 5.2 Task Organization

We ran the shared task in two phases:

**Development Phase** In the first phase, only training and development data were made available, and no gold labels were provided for the latter. The participants competed against each other to achieve the best performance on the development set. A live leaderboard was made available to keep track of all submissions.

**Test Phase** In the second phase, the test set was released and the participants were given just a few days to submit their final predictions.

In the *Development Phase*, the participants could make an unlimited number of submissions, and see the outcome in their private space. The best score for each team, regardless of the submission time, was also shown in a public leaderboard. As a result, not only could the participants observe the impact of various modifications in their systems, but they could also compare against the results by other participating teams. In the *Test Phase*, the participants could again submit multiple runs, but they would not get any feedback on their performance. Only the latest submission of each team was considered as official and was used for the final team ranking. The final leaderboard on the test set was made public after the end of the shared task.

In the *Development Phase*, a total of 15, 10 and 13 teams made at least one submission for ST1, ST2 and ST3, respectively. In the *Test Phase* the number of teams who made official submissions was 16, 8, and 15 for ST1, ST2, ST3, respectively.

After the competition was over, we left the submission system open for the development set, and we plan to reopen it on the test set as well. The up-to-date leaderboards can be found on the website of the competition.<sup>2</sup>

<sup>2</sup><http://propaganda.math.unipd.it/semieval2021task6/>

Rank. Team	Transformers	Models	Repres.	Misc
	BERT RoBERTa XLNet ALBERT DistilBERT DeBERTa	LSTM CNN SVM Naive Bayes Random Forest CRF	Embeddings Char n-grams PoS	Ensemble Data augmentation Postprocessing
1. MinD	☑		☑	☑
2. Alpha	☑		☑	☑
3. Volta	☑		☑	☑
5. AIMH	☑		☑	☑
6. LeCun	☑	☑	☑	☑
7. WVOQ	☑		☑	☑
9. NLyticsFKIE	☑		☑	☑
12. YNU-HPCC	☑	☑	☑	☑
13. CSECUDSG	☑		☑	☑
15. NLP-IITR	☑	☑	☑	☑
1 (Tian et al., 2021)	6 (Dia et al., 2021)	13 (Hossain et al., 2021)		
2 (Feng et al., 2021)	7 (Roele, 2021)	15 (Gupta and Sharma, 2021)		
3 (Gupta et al., 2021)	9 (Pritzkau, 2021)			
5 (Messina et al., 2021)	12 (Zhu et al., 2021)			

Table 2: **ST1**: Overview of the approaches used by the participating systems. ☑=part of the official submission; ✓=considered in internal experiments; *Repres.* stand for Representations. References to system description papers are shown below the table.

## 6 Participants and Results

Below, we give a general description of the systems that participated in the three subtasks and their results, with focus on those ranked among the top-3. Appendix C gives a description of every system.

### 6.1 Subtask 1 (Unimodal: Text)

Table 2 gives an overview of the systems that took part in Subtask 1. We can see that transformers were quite popular, and among them, most commonly used was RoBERTa, followed by BERT. Some participants used learning models such as LSTM, CNN, and CRF in their final systems, while internally, Naïve Bayes and Random Forest were also tried. In terms of representation, embeddings clearly dominated. Moreover, techniques such as ensembles, data augmentation, and post-processing were also used in some systems.

The evaluation results are shown in Table 3, which also includes two baselines: (i) random, and (ii) majority class. The latter always predicts *Loaded Language*, as it is the most frequent technique for Subtask 1 (see Table 1).

The best system **MinD** (Tian et al., 2021) used five transformers: BERT, RoBERTa, XLNet, DeBERTa, and ALBERT. It was fine-tuned on the PTC corpus (Da San Martino et al., 2020a) and then on the training data for Subtask 1.

Rank	Team	F1-Micro	F1-Macro
1	MinD	.593	.290 <sub>2</sub>
2	Alpha	.572	.262 <sub>5</sub>
3	Volta	.570	.266 <sub>3</sub>
4	mmm	.548	.303 <sub>1</sub>
5	AIMH	.539	.245 <sub>6</sub>
6	LeCun	.512	.227 <sub>8</sub>
7	WVOQ	.511	.227 <sub>8</sub>
8	TeamUNCC	.510	.236 <sub>7</sub>
9	NLyticsFKIE	.498	.140 <sub>13</sub>
10	TeiAS	.497	.214 <sub>10</sub>
11	DAJUST	.497	.187 <sub>11</sub>
12	YNUHPCC	.493	.263 <sub>4</sub>
13	CSECUDSG	.489	.185 <sub>12</sub>
14	TeamFPAI	.406	.115 <sub>15</sub>
15	NLPITR	.379	.126 <sub>14</sub>
	<i>Majority baseline</i>	.374	.033
16	TriHeadAttention	.184	.024 <sub>18</sub>
	<i>Random baseline</i>	.064	.044

Table 3: Results for Subtask 1. The systems are ordered by the official score: *F1-micro*.

The final prediction for MinD averages the probabilities for these models, and further uses post-processing rules, e.g., each bigram appearing more than three times is flagged as a *Repetition*.



Team **Alpha** (Feng et al., 2021) was ranked second. However, they used features from images, which was not allowed (images were only allowed for Subtask 3).

Team **Volta** (Gupta et al., 2021) was third. They used a combination of transformers with the [CLS] token as an input to a two-layer feed-forward network. They further used example weighting to address class imbalance.

We should also mention team **LeCun**, which used additional corpora such as the PTC corpus (Da San Martino et al., 2020a), and augmented the training data using synonyms, random insertion/deletion, random swapping, and back-translation.

## 6.2 Subtask 2 (Unimodal: Text)

The approaches for this task varied from modeling it as a question answering (QA) task to performing multi-task learning. Table 4 presents a high-level summary. We can see that BERT dominated, while RoBERTa was much less popular. We further see a couple of systems using data augmentation. Unfortunately, there are too few systems with system description papers for this subtask, and thus it is hard to do a very deep analysis.

Rank.	Team	Trans.	Models	Repres.	Misc							
		BERT	RoBERTa	LSTM	CNN	SVM	ELMo	Pos	Sentiment	Rhetorics	Ensemble	Data augmentation
1.	Volta	✓	✓									
2.	HOMADOS	✓	✓									
3.	TeamFPAI	✓	✓								✓	✓
5.	WVOQ	✓	✓	✓	✓							
6.	CSECUDSG	✓	✓					✓	✓	✓		✓
7.	YNU-HPCC	✓	✓									✓

1 (Gupta et al., 2021) 5 (Roele, 2021)  
 2 (Kaczyński and Przybyła, 2021) 6 (Hossain et al., 2021)  
 3 (Xiaolong et al., 2021) 7 (Zhu et al., 2021)

Table 4: **ST2**: Overview of the approaches used by the participating systems. ✓=part of the official submission; ✓=considered in internal experiments; *Trans.* is for Transformers; *Repres.* is for Representations. References to system description papers are shown below the table.

Table 5 shows the evaluation results. We report our random baseline, which is based on the random selection of spans with random lengths and a random assignment of labels.

Rank	Team	F1	Precision	Recall
1	Volta	<b>.482</b>	.501 <sub>2</sub>	<b>.464<sub>1</sub></b>
2	HOMADOS	.407	.412 <sub>3</sub>	.403 <sub>2</sub>
3	TeamFPAI	.397	<b>.652<sub>1</sub></b>	.286 <sub>5</sub>
4	TeamUNCC	.329	.285 <sub>4</sub>	.390 <sub>3</sub>
5	WVOQ	.268	.243 <sub>5</sub>	.299 <sub>4</sub>
6	CSECUDSG	.120	.080 <sub>8</sub>	.243 <sub>6</sub>
7	YNUHPCC	.091	.186 <sub>6</sub>	.060 <sub>7</sub>
8	TriHeadAttention	.080	.170 <sub>7</sub>	.052 <sub>8</sub>
	<i>Random Baseline</i>	<i>.010</i>	<i>.034</i>	<i>.006</i>

Table 5: Results for Subtask 2. The systems are ordered by the official score: *F1-micro*.

The best model by team **Volta** (Gupta et al., 2021) used various transformer models, such as BERT and RoBERTa, to predict token classes by considering the output of each token embedding. Then, they assigned classes for a given word as the union of the classes predicted for the subwords that make that word (to account for BPEs).

Team **HOMADOS** (Kaczyński and Przybyła, 2021) was second, and they used a multi-task learning (MTL) and additional datasets such as the PTC corpus from SemEval-2020 task 11 (Da San Martino et al., 2020a), and a fake news corpus (Przybyła, 2020). They used BERT, followed by several output layers that perform auxiliary tasks of propaganda detection and credibility assessment in two distinct scenarios: sequential and parallel MTL. Their final submission used the latter.

Team **TeamFPAI** (Xiaolong et al., 2021) formulated the task as a question answering problem using machine reading comprehension, thus improving over the ensemble-based approach of Liu et al. (2018). They further explored data augmentation and loss design techniques, in order to alleviate the problem of data sparseness and data imbalance.

## 6.3 Subtask 3 (Multimodal: Memes)

Table 6 presents an overview of the approaches used by the systems that participated in Subtask 3. This is a very rich and very interesting table. We can see that transformers were quite popular for text representation, with BERT dominating, but RoBERTa being quite popular as well. For the visual modality, the most common representations were variants of ResNet, but VGG16 and CNNs were also used. We further see a variety of representations and fusion methods, which is to be expected given the multi-modal nature of this subtask.

Rank. Team	Transformers	Models	Representations	Fusion	Misc
	BERT RoBERTa XLNet ALBERT FastBERT GPT-2 DeBERTa	ResNet18 ResNet50 ResNet51 VGG16 LSTM CNN SVM CRF	Embeddings ELMo Words/Word n-grams Char n-grams PoS Sentiment Rhetorics FR (ResNet34) MS OCR YouTube-8M CLIP BUTD ERNIE-VIL SemVLP	Average Concat Attention MLP Chained classifier	Ensemble Data augmentation Postprocessing
1. Alpha	✓	✓	✓	✓	✓
2. MinD	✓	✓	✓	✓	✓
3. 1213Li	✓	✓	✓	✓	✓
4. AIMH	✓	✓	✓	✓	✓
5. Volta	✓	✓	✓	✓	✓
6. CSECUDSG	✓	✓	✓	✓	✓
8. LIIR	✓	✓	✓	✓	✓
10. WVOQ	✓	✓	✓	✓	✓
11. YNU-HPCC	✓	✓	✓	✓	✓
13. NLyticsFKIE	✓	✓	✓	✓	✓
15. LT3-UGent	✓	✓	✓	✓	✓
	1 (Feng et al., 2021)	5 (Gupta et al., 2021)	11 (Zhu et al., 2021)		
	2 (Tian et al., 2021)	6 (Hossain et al., 2021)	13 (Pritzkau, 2021)		
	3 (Peiguang et al., 2021)	8 (Ghadery et al., 2021)	15 (Singh and Lefever, 2021)		
	4 (Messina et al., 2021)	10 (Roele, 2021)			



Table 6: **ST3**: Overview of the approaches used by the participating systems. =part of the official submission; =considered in internal experiments. References to system description papers are shown below the table.

Table 7 shows the performance on the test set for the participating systems for Subtask 3. The two baselines shown in the table are similar to those for Subtask 1, namely a random baseline and a majority class baseline. However, this time the most frequent class baseline always predicts *Smears* (for Subtask 1, it was *Loaded Language*), as this is the most frequent technique for Subtask 3 (as can be seen in Table 1).

Team **Alpha** (Feng et al., 2021) pre-trained a transformer using text with visual features. They extracted grid features using ResNet50, and salient region features using BUTD. They further used these grid features to capture the high-level semantic information in the images. Moreover, they used salient region features to describe objects and to caption the event present in the memes. Finally, they built an ensemble of fine-tuned DeBERTa+ResNet, DeBERTa+BUTD, and ERNIE-VIL systems.

Team **MinD** (Tian et al., 2021) combined a system for Subtask 1 with (i) ResNet-34, a face recognition system, (ii) OCR-based positional embeddings for text boxes, and (iii) Faster R-CNN to extract region-based image features. They used late fusion to combine the textual and the visual representations. Other multimodal fusion strategies they tried were concatenation of the representation and mapping using a multi-layer perceptron.

Team **1213Li** (Peiguang et al., 2021) used RoBERTa and ResNet-50 as feature extractors for texts and images, respectively, and adopted a label embedding layer with a multi-modal attention mechanism to measure the similarity between labels with multi-modal information, and fused features for label prediction.

Rank	Team	F1-Micro	F1-Macro
1	Alpha	.581	.273 <sub>1</sub>
2	MinD	.566	.244 <sub>3</sub>
3	1213Li	.549	.228 <sub>5</sub>
4	AIMH	.540	.207 <sub>6</sub>
5	Volta	.521	.189 <sub>8</sub>
6	CSECUDSG	.513	.121 <sub>11</sub>
7	aircasMM	.511	.200 <sub>7</sub>
8	LIIR	.498	.188 <sub>9</sub>
9	CAU731NLP	.481	.084 <sub>14</sub>
10	WVOQ	.478	.240 <sub>4</sub>
11	YNUHPCC	.446	.096 <sub>13</sub>
12	TriHeadAttention	.442	.062 <sub>15</sub>
13	NLyticsFKIE	.423	.118 <sub>12</sub>
	<i>Majority baseline</i>	.354	.036
14	LT3UGent	.332	.264 <sub>2</sub>
15	TeamUNCC	.224	.124 <sub>10</sub>
	<i>Random baseline</i>	.071	.052

Table 7: Results for Subtask 3. The systems are ordered by the official score: *F1-micro*.

## 7 Conclusion and Future Work

We presented *SemEval-2021 Task 6 on Detection of Persuasion Techniques in Texts and Images*. It was a successful task: a total of 71 teams registered to participate, 22 teams eventually made an official submission on the test set, and 15 teams also submitted a task description paper.

In future work, we plan to increase the data size and to add more propaganda techniques. We further plan to cover several different languages.

### Acknowledgments

This research part of the Tanbih mega-project,<sup>3</sup> which is developed at the Qatar Computing Research Institute, HBKU, and aims to limit the impact of “fake news,” propaganda, and media bias by making users aware of what they are reading.

### Ethics and Broader Impact

**User Privacy** Our dataset only includes memes and it contains no user information.

**Biases** Any biases in the dataset are unintentional, and we do not intend to do harm to any group or individual. Note that annotating propaganda techniques can be subjective, and thus it is inevitable that there would be biases in our gold-labeled data or in the label distribution. We address these concerns by collecting examples from a variety of users and groups, and also by following a well-defined schema, which has clear definitions and on which we achieved high inter-annotator agreement.

Moreover, we had a diverse annotation team, which included six members, both female and male, all fluent in English, with qualifications ranging from undergrad to MSc and PhD degrees, including experienced NLP researchers, and covering multiple nationalities. This helped to ensure the quality. No incentives were provided to the annotators.

**Misuse Potential** We ask researchers to be aware that our dataset can be maliciously used to unfairly moderate memes based on biases that may or may not be related to demographics and other information within the text. Intervention with human moderation would be required in order to ensure this does not occur.

<sup>3</sup><http://tanbih.qcri.org/>

## References

- Shamsiah Abd Kadir, Anitawati Lokman, and T. Tsuchiya. 2016. Emotion and techniques of propaganda in YouTube videos. *Indian Journal of Science and Technology*, Vol (9):1–8.
- Shamsiah Abd Kadir and Ahmad Sauffiyani. 2014. A content analysis of propaganda in Harakah newspaper. *Journal of Media and Information Warfare*, 5:73–116.
- Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. YouTube-8M: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*.
- Ivan Habernal et al. 2018. Before name-calling: Dynamics and triggers of ad hominem fallacies in web argumentation. In *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT’18*, pages 386–396, New Orleans, LA, USA.
- Ron Artstein and Massimo Poesio. 2008. Survey article: Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.
- Pepa Atanasova, Lluís Màrquez, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Wajdi Zaghrouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 1: Check-worthiness. In *CLEF 2018 Working Notes*, Avignon, France.
- Pepa Atanasova, Preslav Nakov, Georgi Karadzhov, Mitra Mohtarami, and Giovanni Da San Martino. 2019. Overview of the CLEF-2019 CheckThat! lab on automatic identification and verification of claims. Task 1: Check-worthiness. In *Working Notes of CLEF 2019*, Lugano, Switzerland.
- Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Pepa Atanasova, Wajdi Zaghrouani, Spas Kyuchukov, Giovanni Da San Martino, and Preslav Nakov. 2018. Overview of the CLEF-2018 CheckThat! lab on automatic identification and verification of political claims, task 2: Factuality. In *CLEF 2018 Working Notes*, Avignon, France.
- Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Nikolay Babulkov, Bayan Hamdan, Alex Nikolov, Shaden Shaar, and Zien Sheikh Ali. 2020. Overview of CheckThat! 2020 — automatic identification and verification of claims in social media. In *Proceedings of the 11th International Conference of the CLEF Association: Experimental IR Meets Multilinguality, Multimodality, and Interaction*, CLEF ’2020.

- Alberto Barrón-Cedeño, Tamer Elsayed, Preslav Nakov, Giovanni Da San Martino, Maram Hasanain, Reem Suwaileh, and Fatima Haouari. 2020. CheckThat! at CLEF 2020: Enabling the automatic identification and verification of claims in social media. In *Proceedings of the European Conference on Information Retrieval, ECIR '19*, pages 499–507, Lisbon, Portugal.
- Alberto Barrón-Cedeno, Israa Jaradat, Giovanni Da San Martino, and Preslav Nakov. 2019. Proppy: Organizing the news based on their propagandistic content. *Information Processing & Management*, 56(5):1849–1864.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019a. Findings of the NLP4IF-2019 shared task on fine-grained propaganda detection. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda, NLP4IF '19*, pages 162–170, Hong Kong, China.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020a. SemEval-2020 task 11: Detection of propaganda techniques in news articles. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation, SemEval '20*, pages 1377–1414, Barcelona, Spain.
- Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020b. A survey on computational propaganda detection. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI-PRICAI '20*, pages 4826–4832.
- Giovanni Da San Martino, Shaden Shaar, Yifan Zhang, Seunghak Yu, Alberto Barrón-Cedeno, and Preslav Nakov. 2020c. Prta: A system to support the analysis of propaganda techniques in the news. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL '20*, pages 287–293.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019b. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP '19*, pages 5636–5646, Hong Kong, China.
- Leon Derczynski, Kalina Bontcheva, Maria Liakata, Rob Procter, Geraldine Wong Sak Hoi, and Arkaitz Zubiaga. 2017. SemEval-2017 Task 8: RumourEval: Determining rumour veracity and support for rumours. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval '17*, pages 60–67, Vancouver, Canada.
- Abujaber Dia, Qarqaz Ahmed, and Abdullah Malak A. 2021. LeCun at SemEval-2021 Task 6: Detecting persuasion techniques in text using ensembled pre-trained transformers and data augmentation. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019a. CheckThat! at CLEF 2019: Automatic identification and verification of claims. In *Advances in Information Retrieval, ECIR '19*, pages 309–315, Lugano, Switzerland.
- Tamer Elsayed, Preslav Nakov, Alberto Barrón-Cedeño, Maram Hasanain, Reem Suwaileh, Giovanni Da San Martino, and Pepa Atanasova. 2019b. Overview of the CLEF-2019 CheckThat!: Automatic identification and verification of claims. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction, LNCS*, pages 301–321.
- Zhida Feng, Jiji Tang, Jiayang Liu, Weichong Yin, Shikun Feng, Yu Sun, and Li Chen. 2021. Alpha at SemEval-2021 Tasks 6: Transformer based propaganda classification. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Erfan Ghadery, Damien Sileo, and Marie-Francine Moens. 2021. LIIR at SemEval 2021 Task 6: Detection of persuasion techniques in texts and images using CLIP features. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Maria Glenski, E. Ayton, J. Mendoza, and Svitlana Volkova. 2019. Multilingual multimodal digital deception detection and disinformation spread across social platforms. *ArXiv*, abs/1909.05838.
- Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours. In *Proceedings of the 13th International Workshop on Semantic Evaluation, SemEval '19*, pages 845–854, Minneapolis, Minnesota, USA.
- Bin Guo, Yasan Ding, Lina Yao, Yunji Liang, and Zhiwen Yu. 2020. The future of false information detection on social media: New perspectives and trends. *ACM Comput. Surv.*, 53(4).
- Kshitij Gupta, Devansh Gautam, and Radhika Mamidi. 2021. Volta at SemEval-2021 Task 6: Towards detecting persuasive texts and images using textual and multimodal ensemble. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Vansh Gupta and Raksha Sharma. 2021. NLPITR at SemEval-2021 Task 6: detection of persuasion techniques in texts and images. In *Proceedings of the International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.

- Ivan Habernal, Raffael Hannemann, Christian Polak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. Argotario: Computational argumentation meets serious games. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 7–12, Copenhagen, Denmark.
- Ivan Habernal, Patrick Pauli, and Iryna Gurevych. 2018. Adapting serious game for fallacious argumentation to German: Pitfalls, insights, and best practices. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, LREC'18, Miyazaki, Japan.
- Maram Hasanain, Fatima Haouari, Reem Suwaileh, Zien Sheikh Ali, Bayan Hamdan, Tamer Elsayed, Alberto Barrón-Cedeño, Giovanni Da San Martino, and Preslav Nakov. 2020. Overview of CheckThat! 2020 Arabic: Automatic identification and verification of claims in social media. In *Working Notes of CLEF 2020—Conference and Labs of the Evaluation Forum*, CLEF '2020.
- Maram Hasanain, Reem Suwaileh, Tamer Elsayed, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. Overview of the CLEF-2019 CheckThat! Lab on Automatic Identification and Verification of Claims. Task 2: Evidence and Factuality. In *CLEF 2019 Working Notes. Working Notes of CLEF 2019 - Conference and Labs of the Evaluation Forum*, Lugano, Switzerland. CEUR-WS.org.
- Tashin Hossain, Jannatun Naim, Faren Tasneem, Radiathun Tashia, and Abu Nowshed Chy. 2021. CSE-CUDSG at SemEval-2021 Task 6: Orchestrating multimodal neural architectures for identifying persuasion techniques in texts and images. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Konrad Kaczyński and Piotr Przybyła. 2021. HOMA-DOS at SemEval-2021 Task 6: Multi-task learning for propaganda detection. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Douwe Kiela, Suvrat Bhooshan, Hamed Firooz, and Davide Testuggine. 2019. Supervised multimodal bitransformers for classifying images and text. In *Proceedings of the NeurIPS 2019 Workshop on Visually Grounded Interaction and Language*, ViGIL@NeurIPS '19.
- Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. The hateful memes challenge: Detecting hate speech in multimodal memes. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, NeurIPS '20.
- J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.
- Jiahua Liu, Wan Wei, Maosong Sun, Hao Chen, Yantao Du, and Dekang Lin. 2018. A multi-answer multi-task framework for real-world machine reading comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, EMNLP '18, pages 2109–2118, Brussels, Belgium.
- Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. FastBERT: a self-distilling BERT with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, ACL '20, pages 6035–6044, Online.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proceedings of the Conference on Neural Information Processing Systems*, NeurIPS '19, pages 13–23, Vancouver, Canada.
- Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020. A survey on computational propaganda detection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, IJCAI-PRICAI '20, pages 4826–4832.
- Nicola Messina, Fabrizio Falchi, Claudio Gennaro, and Giuseppe Amato. 2021. AIMH at SemEval-2021 Task 6: multimodal classification using an ensemble of transformer models. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, SemEval '19, pages 860–869, Minneapolis, Minnesota, USA.
- Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations, Workshop Track*, ICLR '13, Scottsdale, Arizona, USA.
- Clyde R. Miller. 1939. The Techniques of Propaganda. From “How to Detect and Analyze Propaganda,” an address given at Town Hall. The Center for learning.
- Preslav Nakov, Alberto Barrón-Cedeño, Tamer Elsayed, Reem Suwaileh, Lluís Màrquez, Wajdi Zaghouani, Pepa Atanasova, Spas Kyuchukov, and Giovanni Da San Martino. 2018. Overview of the

- CLEF-2018 CheckThat! lab on automatic identification and verification of political claims. In *Proceedings of the International Conference of CLEF*, CLEF '18, pages 372–387, Avignon, France.
- Preslav Nakov, Giovanni Da San Martino, Tamer Elsayed, Alberto Barrón-Cedeño, Rubén Míguez, Shaden Shaar, Firoj Alam, Fatima Haouari, Maram Hasanain, Nikolay Babulkov, Alex Nikolov, Gautam Kishore Shahi, Julia Maria Struß, and Thomas Mandl. 2021. The CLEF-2021 CheckThat! lab on detecting check-worthy claims, previously fact-checked claims, and fake news. In *Advances in Information Retrieval*, ECIR '21, pages 639–649.
- Li Peiguang, Li Xuan, and Sun Xian. 2021. 1213Li at SemEval-2021 Task 6: detection of propaganda with multi-modal attention and pre-trained models. In *Proceedings of the Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. 2010. An Evaluation Framework for Plagiarism Detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING'10, pages 997–1005, Beijing, China.
- Albert Pritzkau. 2021. NLyticsFKIE at SemEval-2021 Task 6: Detection of persuasion techniques in texts and images. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Piotr Przybyla. 2020. Capturing the style of fake news. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):490–497.
- Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '17, pages 2931–2937, Copenhagen, Denmark.
- Cees Roele. 2021. WVOQ at SemEval-2021 Task 6: BART for span detection and classification. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Hyunjin Seo. 2014. [Visual propaganda in the age of social media: An empirical analysis of Twitter images during the 2012 Israeli– Hamas conflict](#). *Visual Communication Quarterly*, 21(3):150–161.
- Shaden Shaar, Firoj Alam, Giovanni Da San Martino, Alex Nikolov, Wajdi Zaghouni, and Preslav Nakov. 2021. Findings of the NLP4IF-2021 shared task on fighting the COVID-19 infodemic. In *Proceedings of the Fourth Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, NLP4IF@NAACL' 21.
- Shaden Shaar, Alex Nikolov, Nikolay Babulkov, Firoj Alam, Alberto Barrón-Cedeño, Tamer Elsayed, Maram Hasanain, Reem Suwaileh, Fatima Haouari, Giovanni Da San Martino, and Preslav Nakov. 2020. Overview of CheckThat! 2020 English: Automatic identification and verification of claims in social media. In *Working Notes of CLEF 2020—Conference and Labs of the Evaluation Forum*, CLEF '2020.
- Anup Shah. 2005. War, propaganda and the media.
- Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *SIGKDD Explor. Newsl.*, 19(1):22–36.
- Pranaydeep Singh and Els Lefever. 2021. LT3 at SemEval-2021 Task 6: Using multi-modal compact bilinear pooling to combine visual and textual understanding in memes. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL-HLT '18, pages 809–819, New Orleans, Louisiana.
- Junfeng Tian, Min Gui, Chenliang Li, Ming Yan, and Wenming Xiao. 2021. MinD at SemEval-2021 Task 6: Propaganda detection using transfer learning and multimodal fusion. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Robyn Torok. 2015. Symbiotic radicalisation strategies: Propaganda tools and neuro linguistic programming. In *Proceedings of the Australian Security and Intelligence Conference*, ASIC '15, pages 58–65, Perth, Australia.
- Svitlana Volkova, Ellyn Ayton, Dustin L. Arendt, Zhuanyi Huang, and Brian Hutchinson. 2019. Explaining multimodal deceptive news prediction models. In *Proceedings of the International Conference on Web and Social Media*, ICWSM '19, pages 659–662, Munich, Germany.
- Anthony Weston. 2008. *A rulebook for arguments*. Hackett Publishing.
- Hou Xiaolong, Ren Junsong, Rao Gang, Jiang Lianxin, Ruan Zhihao, Yang Mo, and Shen Jianping. 2021. TeamFPAI at SemEval-2020 Task 6: BERT-MRC for propaganda techniques detection. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.
- Xingyu Zhu, Jin Wang, and Xuejie Zhang. 2021. YNU-HPCC at SemEval-2021 Task 6: Combining ALBERT and Text-CNN for persuasion detection in texts and images. In *Proceedings of the International Workshop on Semantic Evaluation*, SemEval '21, Bangkok, Thailand.

## Appendix

### A Data Collection and Annotation

#### A.1 Data Collection

To collect the data for the dataset, we used Facebook, as it has many public groups with a large number of users, who intentionally or unintentionally share a large number of memes. We used our own private Facebook accounts to crawl the public posts from users and groups. To make sure the resulting feed had a sufficient number of memes, we initially selected some public groups focusing on topics such as politics, vaccines, COVID-19, and gender equality. Then, using the links between groups, we expanded our initial group pool to a total of 26 public groups. We went through each group, and we collected memes from old posts, dating up to three months before the newest post in the group. Out of the 26 groups, 23 were about politics, US and Canadian: left, right, centered, anti-government, and gun control. The other 3 groups were on general topics such as health, COVID-19, pro-vaccines, anti-vaccines, and gender equality. Even though the number of political groups was larger (i.e., 23), the other 3 general groups had a higher number of users and a substantial amount of memes.

#### A.2 Annotation Process

We annotated the memes using the 22 persuasion techniques from Section 3 in a multi-label setup. Our annotation focused (i) on the text only, using 20 techniques, and (ii) on the entire meme (text + image), using all 22 techniques.

We could not annotate the visual modality as an independent task because memes have the text as part of the image. Moreover, in many cases, the message in the meme requires both modalities. For example, in Figure 28, the image by itself does not contain any persuasion technique, but together with the text, we can see *Smears* and *Reductio at Hitlerum*.

The annotation team included six members, both female and male, all fluent in English, with qualifications ranging from undergrad to MSc and PhD degrees, including experienced NLP researchers, and covering multiple nationalities. This helped to ensure the quality of the annotation, and our focus was really on having very high-quality annotation. No incentives were given to the annotators.

We used PyBossa<sup>4</sup> as an annotation platform, as it provides the functionality to create a custom annotation interface that we found to be a good fit for our needs in each phase of the annotation process. Figure 4 shows examples of the annotation interface for the five different phases of annotation, which we describe in detail below.

**Phase 1: Filtering and Text Editing** The first phase of the annotation process is about selecting the memes for our task, followed by extracting and editing the textual contents of each meme. After we collected the memes, we observed that we needed to remove some of them as they did not fit our definition: “*photograph style image with a short text on top of it.*” Thus, we asked the annotators to exclude images with the characteristics listed below. During this phase, we filtered out a total of 111 memes.

- Images with diagrams/graphs/tables (see Figure 5a).
- Cartoons. (see Figure 5b)
- Memes for which no multi-modal analysis is possible: e.g., only text, only image, etc. (see Figure 5c)

Next, we used the Google Vision API<sup>5</sup> to extract the text from the memes. As the resulting text sometimes contains errors, manual checking was needed to correct it. Thus, we defined several text editing rules, and we asked the annotators to apply them on the memes that passed the filtering rules above.

1. When the meme is a screenshot of a social network account, e.g., WhatsApp, the user name and login can be removed as well as all “Like”, “Comment”, “Share”.
2. Remove the text related to logos that are not part of the main text.
3. Remove all text related to figures and tables.
4. Remove all text that is partially hidden by an image, so that the sentence is almost impossible to read.
5. Remove all text that is not from the meme, but on banners carried on by demonstrators, street advertisements, etc.

<sup>4</sup><https://pybossa.com>

<sup>5</sup><http://cloud.google.com/vision>

### Phase 1: Memes text editing project: Contribute

Search | 3:57 AM Faye Maureen Hearn Yesterday at 3:51 PM A journalist asked Justin Trudeau when more help would arrive for seniors... Souran Globni News CANADA.CA/CORONAVIRUS he answered by saying "the priority was on helping other groups of Canadians first." ORobert Dasher and 39 others 18 Comments 0 Like Comment Send

Please edit the text:

Search | 3:57 AM Faye Maureen Hearn Yesterday at 3:51 PM A journalist asked Justin Trudeau when more help would arrive for seniors... Souran Globni News CANADA.CA/CORONAVIRUS he answered by saying "the priority was on helping other groups of Canadians first." ORobert Dasher and 39 others 18 Comments 0 Like Comment Send

Editing Guidelines

Submit Reject

### Phase 2: Propaganda techniques annotation in text: Contribute

Text to perform selection on:

HARRY S. TRUMAN  
"YOU CAN'T GET RICH IN POLITICS UNLESS YOU'RE A CROOK."  
WE KNOW

Techniques in the text

- Loaded Language
- Appeal to fear/prejudice
- Name calling/Labeling
- Flag-waving
- Doubt
- Exaggeration/Minimisation
- Slogans
- Causal Oversimplification
- Thought-terminating cliché
- Appeal to authority
- Black-and-white Fallacy/Dictatorship
- Reductio ad hitlerum
- Whataboutism
- Presenting Irrelevant Data (Red Herring)
- Misrepresentation of Someone's Position (Straw Man)
- Obfuscation, Intentional vagueness, Confusion
- Bandwagon
- Repetition
- Smears
- Glittering generalities (Virtue)

Currently selected:

["start": 0, "end": 71, "technique": "Appeal to authority", "text": "HARRY S. TRUMAN('YOU CAN'T GET RICH IN POLITICS UNLESS YOU'RE A CROOK,')"]

["start": 73, "end": 80, "technique": "Thought-terminating cliché", "text": "WE KNOW"]

["start": 64, "end": 69, "technique": "Loaded Language", "text": "CROOK"]

Definitions of propaganda techniques in text

Submit

### Phase 3: Consolidation of techniques in the text: Contribute

Text to perform selection on:

"Socialism only works in two places: Heaven where they don't need it, and hell where they already have it."  
Ronald Regan

Notes:

Techniques in the text

- Loaded Language
- Appeal to fear/prejudice
- Name calling/Labeling
- Flag-waving
- Doubt
- Exaggeration/Minimisation
- Slogans
- Causal Oversimplification
- Thought-terminating cliché
- Appeal to authority
- Black-and-white Fallacy/Dictatorship
- Reductio ad hitlerum
- Whataboutism
- Presenting Irrelevant Data (Red Herring)
- Misrepresentation of Someone's Position (Straw Man)
- Obfuscation, Intentional vagueness, Confusion
- Bandwagon
- Repetition
- Smears
- Glittering generalities (Virtue)

Currently selected:

["start": 0, "end": 120, "technique": "Appeal to authority", "text": "Socialism only works in two places: Heaven where they don't need it, and hell where they already have it."]

["start": 1, "end": 10, "technique": "Loaded Language", "text": "Socialism"]

["start": 1, "end": 106, "technique": "Black-and-white Fallacy/Dictatorship", "text": "Socialism only works in two places: Heaven where they don't need it, and hell where they already have it."]

["start": 37, "end": 43, "technique": "Loaded Language", "text": "Heaven"] (2)

["start": 74, "end": 78, "technique": "Loaded Language", "text": "hell"] (2)

["start": 108, "end": 120, "technique": "Appeal to authority", "text": "Ronald Regan"]

Definitions of propaganda techniques in text

Submit

### Phase 4: Propaganda technique annotation in images: Contribute

Techniques in the text

- Loaded Language
- Appeal to fear/prejudice
- Name calling/Labeling
- Flag-waving
- Doubt
- Exaggeration/Minimisation
- Slogans
- Causal Oversimplification
- Thought-terminating cliché
- Appeal to authority
- Black-and-white Fallacy/Dictatorship
- Reductio ad hitlerum
- Whataboutism
- Presenting Irrelevant Data (Red Herring)
- Misrepresentation of Someone's Position (Straw Man)
- Obfuscation, Intentional vagueness, Confusion
- Bandwagon
- Repetition
- Smears
- Glittering generalities (Virtue)

Techniques in the image

- Transfer
- Appeal to (Strong) Emotions

ANTIFA "A courageous group of Americans." - Joe Biden (Apr 23, 2019)

Definitions of propaganda techniques

Submit

### Phase 5: Image Consolidation (Final): Contribute

Techniques in the text

- Loaded Language
- Appeal to fear/prejudice (1)
- Name calling/Labeling
- Flag-waving
- Doubt
- Exaggeration/Minimisation
- Slogans
- Causal Oversimplification
- Thought-terminating cliché
- Appeal to authority
- Black-and-white Fallacy/Dictatorship
- Reductio ad hitlerum
- Whataboutism
- Presenting Irrelevant Data (Red Herring)
- Misrepresentation of Someone's Position (Straw Man)
- Obfuscation, Intentional vagueness, Confusion
- Bandwagon
- Repetition
- Smears (3)
- Glittering generalities (Virtue)

Techniques in the image

- Transfer (1)
- Appeal to (Strong) Emotions (3)

TRUMP BODY COUNT  
September 16, 2020  
200,770  
TRUMPS IDEA OF WINNING

TRUMP BODY COUNT September 16, 2020 200,770 TRUMPS IDEA OF WINNING TRUMP -2020- KEEP AMERICA GREAT

Notes from text:

Submit

Figure 4: Examples of the annotation interface for different phases.

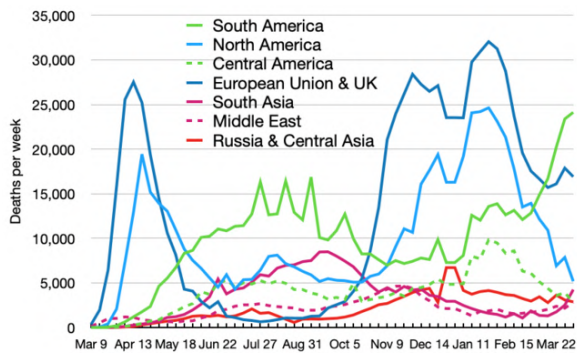
- Remove the author of the meme if it is signed.
- If the text is in columns, first put all text from the first column, then all text from the next column, etc.
- Rearrange the text, so that there is one sentence per line, whenever possible.
- If there are separate blocks of text in different locations of the image, separate them by a blank line. However, if it is evident that the text blocks are part of a single sentence, keep them together.

**Phase 2: Text Annotation** The annotations for phase 2 are targeted at Subtasks 1 and 2. Given the list of propaganda techniques for text only annotation, as discussed in Section A.4 (i.e., techniques 1-20), and the textual content of the target meme, the annotators were asked to identify which techniques appear in the text, and also to annotate the span of each instance of a technique use. In this phase, there were three annotators per example.

**Phase 3: Text Consolidation** Phase 3 is the consolidation step for the annotations from phase 2. The three annotators met with the rest of the team, who acted as consolidators, and discussed each annotation, so that a consensus could be reached.



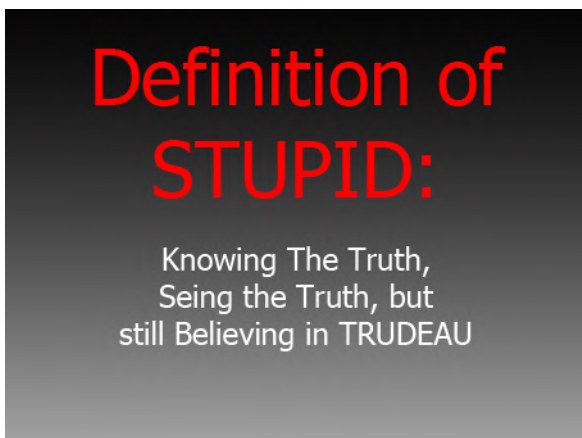
**As some #European countries refuse full lockdown the deaths per week are skyrocketing.**



(a) Example of a meme with a **graph** *Source(s):* Image ; License



(b) Example of a **cartoon** meme; *Source(s):* Image ; License



(c) Example of a meme with **only text** modality; License .

Figure 5: Examples of memes we filtered out.

We made sure to consider different interpretations and to anotate techniques corresponding to the most likely one. While this phase was devoted to checking the annotations from phase 2, when a novel instance of a technique was found, it could be added; conversely, an instance of a technique with perfect agreement from phase 2 could also be dropped. Phase 3 was essential for ensuring quality, and it served as an additional training opportunity for the entire team, which was very useful.

**Phase 4: Multimodal Annotation** In this phase, the goal is to identify which of the 22 techniques, discussed in Section A.4, appear in the meme: in the text and in the visual content. Note that some of the techniques occurring in the text might be identified only in this phase because the image provides the necessary context. Thus, we presented the meme with the consolidated propaganda labels from phase 3. We intentionally provided the consolidated text labels to the annotators in order to ensure that they focus their attention on identifying propaganda techniques that require both modalities rather than repeating what was already labeled in the earlier phases. In this phase, there were three annotators per example.

**Phase 5: Multimodal Consolidation.** In phase 5, we consolidated the annotations from phase 4 in a discussion of the entire team of six annotators (just as we did for phase 3).

### A.3 Annotation Agreement

We assessed the quality for the individual annotators from phases 2 and 4 (i.e., when combining the annotations for the meme’s text and for the entire meme) to the final consolidated labels at phase 5. Since our annotation is multi-label, we computed Krippendorff’s  $\alpha$  (Artstein and Poesio, 2008). The results are shown in Table 8, and the numbers indicate moderate to substantial agreement (Landis and Koch, 1977).

Agreement Pair	Krippendorff’s $\alpha$
Annotator 1 vs. Consolidated	0.83
Annotator 2 vs. Consolidated	0.91
Annotator 3 vs. Consolidated	0.56
<b>Average</b>	<b>0.77</b>

Table 8: Inter-annotator agreement in terms of Krippendorff’s  $\alpha$  between each of the annotators and the consolidated annotation.

### A.4 Propaganda Techniques: Definitions

Below, we present the definitions of our 22 propaganda techniques, together with examples: both textual, and memes. Note that, for copyright reasons, we show our own recreated versions of actual memes from our dataset, where, for each meme, we indicate the image(s) we used and the corresponding license terms (as hyperlinks in the image caption).

**1. Loaded Language:** Using specific words and phrases with strong emotional implications (i.e., either positive or negative) to influence an audience.

An example meme is shown in Figure 6, which contains four instances of this persuasion technique in its text: *killed thousands of innocents*, *retaliate*, *kill*, and *warmonger*.



Figure 6: Example for **Loaded Language**; *Source(s)*: Image 1, Image 2; License 1, License 2

**2. Name Calling or Labeling:** Labeling the object of the propaganda as either something the target audience fears, hates, finds undesirable, or loves, praises.

Figure 7 shows three instances of this technique: *the two biggest threats to America*, *the worst senate leader ever*, and *the most corrupt President ever*. Figure 6 also contains an instance: *warmonger*.

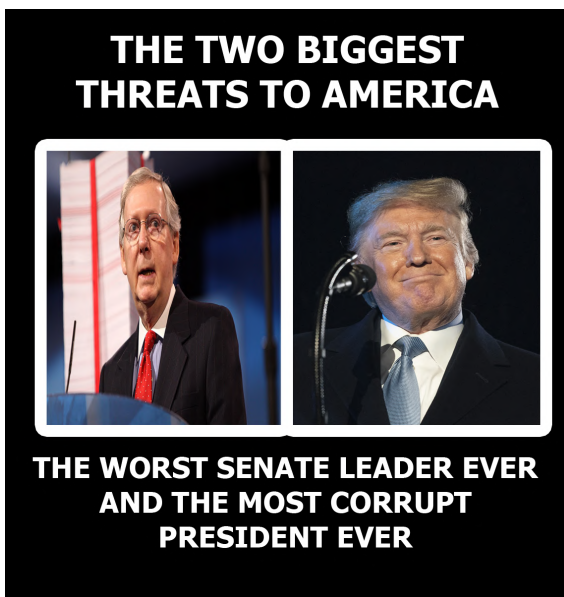


Figure 7: Example for **Name Calling**; *Source(s)*: Image 1, Image 2; License 1, License 2

**3. Doubt:** Questioning the credibility of someone or something.

An example is shown in Figure 8, where the entire text in the meme represents a span for this technique, while the image is just for illustration.

Reminder: This building in Oklahoma City was blown up and destroyed just 4 days before Hilary Clinton was to be indicted in the Whitewater scandal. All documents lost!



Figure 8: Example for **Doubt**; *Source(s)*: Image ; License

**4. Exaggeration or Minimisation:** Representing something in an excessive manner, making it larger, better, worse (e.g., *the best of the best*); or making it seem less important or smaller than it really is (e.g., saying that an insult was just a joke).

An example is shown in Figure 9, where the entire meme conveys an exaggeration. Moreover, all three *Name Calling* instances in Figure 7 are also examples of *Exaggeration*.



Figure 9: Example for **Exaggeration**; *Source(s)*: Image ; License

**5. Appeal to Fear/Prejudice:** Seeking to build support for an idea by instilling anxiety and/or panic in the population towards an alternative. In some cases, the support is built based on preconceived judgments.

An example is shown in Figure 10, where both the text and the image instill fear.



Figure 10: Example for **Appeal to Fear**; *Source(s):* Image ; License

**6. Slogans:** A brief and striking phrase that may include labeling and stereotyping. Slogans tend to act as emotional appeals.

An example is shown in Figure 11, which contains a slogan in its textual content: “*Vaccines. It isn’t always about you.*”



Figure 11: Example for **Slogan**; *Source(s):* Image ; License

**7. Whataboutism:** A technique that attempts to discredit an opponent's position by charging them with hypocrisy without directly disproving their argument.

An example meme is shown in Figure 12, where the entire text represents a span for this technique, while the image is just for illustration.



Figure 12: Example for Whataboutism; *Source(s):* Image ; License

**8. Flag-Waving:** Playing on strong national feeling (or to any group such as race, gender, political preference) to justify or promote an action or idea.

An example is shown in Figure 13, with the technique expressed in the text and the image.



Figure 13: Example for Flag-Waving; *Source(s):* Image ; License

**9. Misrepresentation of Someone’s Position (Straw Man):** An opponent’s proposition is substituted with a similar one, which is then refuted in place of the original proposition.

An example meme is shown in Figure 14, which contains an instance of this technique in its text: here, the entire text in the meme represents a span for this technique, while the image is irrelevant for that technique (however, it is relevant for other techniques such as *Smears*).



Figure 14: Example for Misrepresentation of Someone’s Position (Straw Man); Source(s): Image ; License

**10. Causal Oversimplification:** Assuming a single cause or reason when there are actually multiple causes for an issue. It includes transferring blame to one person or group of people without investigating the complexities of the issue.

An example meme is shown in Figure 15, which contains an instance of this technique in its text: “*You can’t get rich in politics unless you are a crook.*” This statement says that if somebody got rich in politics, the only reason for this happening should be that this person is a crook, while in reality there are typically multiple causes. The image is irrelevant for that technique (however, it is relevant for other techniques such as *Smears*).



Figure 15: Example for Causal Oversimplification; Source(s): Image 1, Image 2; License 1, License 2

**11. Appeal to Authority:** Stating that a claim is true simply because a valid authority or expert on the issue said it was true, without any other supporting evidence offered. We consider the special case in which the reference is not an authority or an expert in this technique, although it is referred to as *Testimonial* in literature.

An example meme is shown in Figure 16, which contains a quote by the 3rd President of the United States.

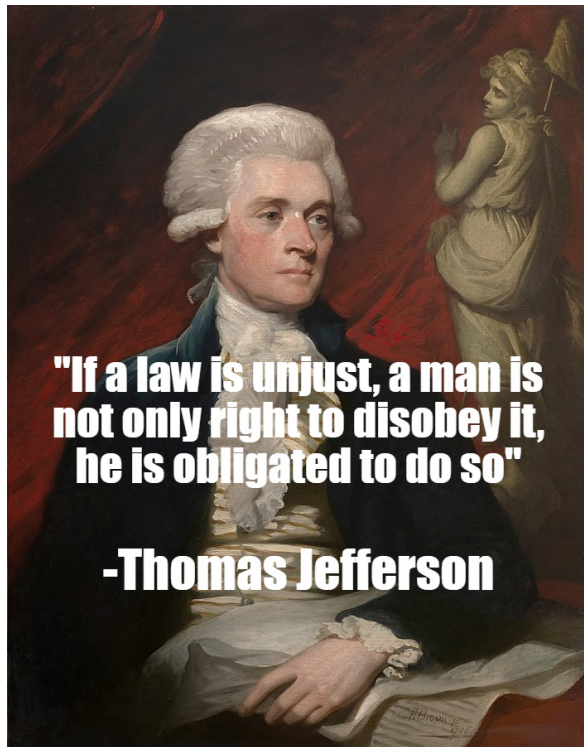


Figure 16: Example for **Appeal to Authority**; *Source(s)*: Image ; License

**12. Thought-Terminating Cliché:** Words or phrases that discourage critical thought and meaningful discussion about a given topic. They are typically short, generic sentences that offer seemingly simple answers to complex questions or that distract attention away from other lines of thought.

Figure 17 shows a meme with an instance of this technique in its text: “*PERIOD.*”



Figure 17: Example for **Thought-Terminating Cliché**; *Source(s)*: Image 1, Image 2; License 1, License 2

**13. Black-and-White Fallacy:** Presenting two alternative options as the only possibilities, when in fact more possibilities exist. We also include dictatorship, where one tells the audience exactly what actions to take, eliminating any other choices.

An example of this technique is shown in Figure 18, which offers only two choices.



Figure 18: Example for **Black-and-White Fallacy**; *Source(s)*: Image 1, Image 2; License 1, License 2

**14. Reductio ad Hitlerum:** Persuading an audience to disapprove an action or idea by suggesting that the idea is popular with groups hated or in contempt by the target audience. It can refer to any person or concept with a negative connotation.

Figure 19 shows a meme trying to discredit the idea of being anti-union by saying that so is Donald Trump, who in turn is shown in bad light.



Figure 19: Example for **Reduction ad Hitlerum**; *Source(s): Image , License*

**15. Repetition:** Repeating the same message, so that the audience eventually accepts it.

An example is shown in Figure 20, where the repetition has a clear rhetorical function.



Figure 20: Example for **Repetition**; *Source(s): Image 1, Image 2, Image 3, Image 4; License 1, License 2, License 3, License 4*

**16. Obfuscation, Intentional Vagueness, Confusion:** Using words that are deliberately unclear, so that the audience may have their own interpretations.

Figure 21, shows an example, where the entire quote by Joe Biden is a span of this technique, as it is unclear what exactly is meant here.



Figure 21: Example for **Obfuscation, Intentional vagueness, Confusion**; *Source(s): Image ; License*

**17. Presenting Irrelevant Data (Red Herring):** Introducing irrelevant material to the issue being discussed, so that everyone’s attention is diverted away from the points made.

An example meme is shown in Figure 22, which contains an instance of this technique in its text. We can see that there is no real connection between the two sentences. Here, the entire text represents a span for this technique, while the image is for reinforcement.

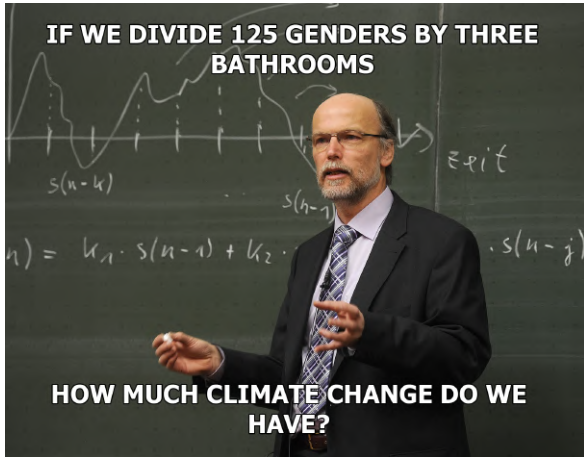


Figure 22: Example for **Presenting Irrelevant Data (Red Herring)**; *Source(s):* Image ; License

**18. Bandwagon:** Attempting to persuade the target audience to join in and take the course of action because “everyone else is taking the same action.”

Figure 23 shows an example that covers the entire text; the image less relevant.



Figure 23: Example for **Bandwagon**; *Source(s):* Image ; License

**19. Smears:** A smear is an effort to damage or to call into question someone’s reputation, by propounding negative propaganda. It can be applied to individuals or groups.

An example meme is shown in Figure 24, where the combination of the image and the text conveys the idea that Biden is unpopular.

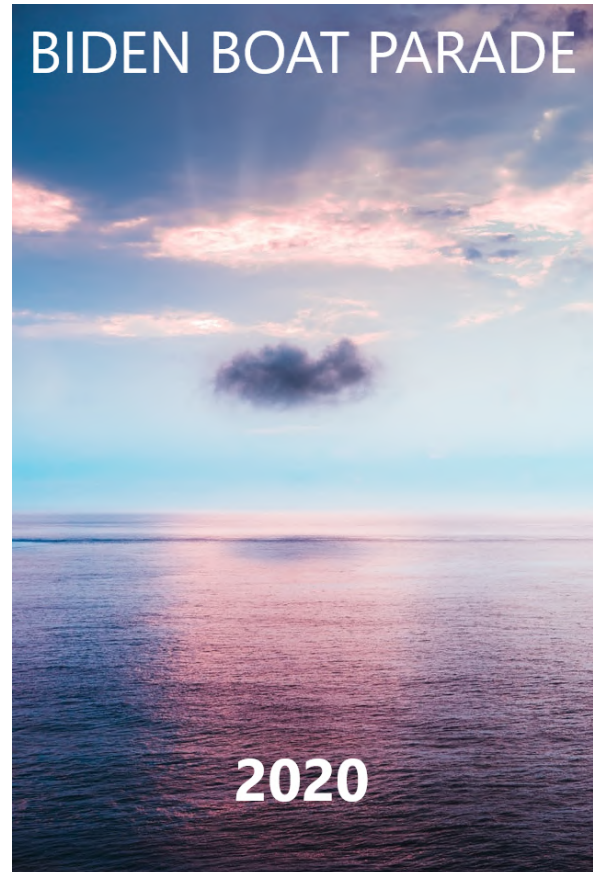


Figure 24: Example for **Smears**; *Source(s):* Image ; License



**20. Glittering Generalities:** These are words or symbols in the value system of the target audience that produce a positive image when attached to a person or issue. Peace, hope, happiness, security, wise leadership, freedom, “The Truth”, etc. are virtue words. Virtue can be also expressed in images, where a person or an object is depicted positively.

Figure 25 shows an example of the use of this technique, in the right half of the meme. The technique covers the entire text span starting from “2 & 1/2 years . . .” until “GDP up 3.2% . . .” It is also expressed in the image, which depicts Donald Trump in a positive way. The text–image combination further strengthens the technique.



Figure 25: Example for **Glittering Generalities**; *Source(s):* Image 1, Image 2; License 1, License 2

**21. Appeal to (Strong) Emotions:** Using images with strong positive/negative emotional implications to influence an audience. We reserve this technique to the images content only.

An example is shown in Figure 26, which invokes strong emotions in the audience.



Figure 26: Example for **Appeal to (Strong) Emotions**; *Source(s):* Image ; License

**22. Transfer:** Also known as *Association*, this is a technique of projecting positive or negative qualities (praise or blame) of a person, entity, object, or value onto another one to make the second one more acceptable or to discredit it. It evokes an emotional response, which stimulates the target to identify with recognized authorities. Often highly visual, this technique often utilizes symbols (for example, the swastikas used in Nazi Germany, originally a symbol for health and prosperity) superimposed over other visual images.

Figure 27 shows an example, where the *Transfer* technique makes use of a communist symbol (namely, hammer and sickle) on top of the pictures of two targeted politicians, with the aim of depicting them in a negative way. The technique is further reinforced by the use of the red color (which is also a symbol of Communism), and by the two instances of *Name Calling* (“*Moscow Mitch*” and “*Moscow’s bitch*”), which make a connection to Moscow (which in turn was the capital of the former Communist block).



Figure 27: Example for **Transfer**; *Source(s)*: Image 1, Image 2; License 1, License 2

## B Subtasks: Definition, Data Format, and Data Examples

Below, we describe the three subtasks and the general data format for each of them. We further show an example of an annotated example for each subtask.

### B.1 Subtask 1

This is a multi-label classification problem, defined as follows:

**Subtask 1 (ST1)** Given only the “textual content” of a meme, identify which of the 20 techniques are used in it.

The data for ST1 comes as a JSON object in the following format:

```
{
  id -> example identifier,
  labels -> list of persuasion
           techniques,
  text -> text of the meme
}
```

Here is an example:

```
{
  "id": "125",
  "labels": [
    "Loaded Language",
    "Name calling/Labeling"
  ],
  "text": "I HATE TRUMP\n\n
          MOST TERRORIST DO"
}
```

### B.2 Subtask 2

ST2 is a more complex version of ST1, as it asks not only for the techniques but also for the exact spans of use each technique. This subtask is a combination of the two subtasks in *SemEval-2020 task 11*. It is a multi-label sequence tagging problem, defined as follows:

**Subtask 2 (ST2)** Given only the “textual content” of a meme, identify which of the 20 techniques are used in it together with the span(s) of text covered by each technique.

The data for ST2 comes as a JSON object with the following format:

```
{
  id -> example identifier,
  text -> text of the meme
  labels : [ -> list of objects
    {
      start -> start index,
      end -> end index,
      technique -> technique,
      text_fragment -> text
    }
  ]
}
```

Here is an example:

```
{
  "id": "125",
  "text": "I HATE TRUMP\n\n
          MOST TERRORIST DO"
  "labels": [
    {
      "start": 2,
      "end": 6,
      "technique": "Loaded Language",
      "text_fragment": "HATE"
    },
    {
      "start": 19,
      "end": 28,
      "technique": "Name calling/
Labeling",
      "text_fragment": "TERRORIST"
    }
  ]
}
```

Note that the labels to be predicted for ST2 are the same ones as for ST1, but this time the spans are to be predicted as well.

### B.3 Subtask 3

ST3 is a multi-modal version of ST1, where the image is also provided. It is a multi-label classification problem, defined as follows:

**Subtask 3 (ST3)** Given a meme, identify which of the 22 techniques are used both in the textual and in the visual content of the meme.

The data for ST3 comes as a JSON object with the following format:

```
{
  id -> example identifier,
  labels -> list of persuasion
           techniques,
  image -> name of the image file,
  text -> text of the meme
}
```

Here is an example:

```
{
  "id": "125",
  "labels": [
    "Loaded Language",
    "Name calling/Labeling",
    "Reductio ad hitlerum",
    "Smears",
  ],
  "image": "125_image.png"
}
```

Here, the image, which is shown in Figure 28), gives rise to two additional persuasion techniques compared to ST1: *Reductio ad Hitlerum* and *Smears*. These techniques are not clearly present in the text alone. Indeed, the image is needed for us to see that there is *Smears*, as this can be only seen when we understand that this is a dialog with a negative propaganda targeting one of the participants (Ilhan Omar). Similarly, we need the image for *Reductio ad Hitlerum*: the image shows us that Ilhan Omar is depicted as a bad person (she is targeted by the *Name Calling* “terrorist”, and she is also the target of the *Smears*), and thus the message being conveyed is that any choice that such a bad person does has to be a bad choice, i.e., hating Trump is a bad thing to do as this is something terrorists do.



Figure 28: The meme with id=125; *Source(s)*: Image 1, Image 2; License 1, License 2

## C Participating Systems

Below, we give a brief description of the participating systems, listed in alphabetical order, with reference to the corresponding task description paper. The numbers in square brackets refer to the official ranking of the target system on the individual subtasks.

**1213Li (Peiguang et al., 2021)[ST3: 3rd]** used RoBERTa and ResNet-50 as feature extractors for texts and images. They used a label embedding layer with a multi-modal attention mechanism to measure the similarity between labels with the multi-modal information and fused features for label prediction.

**AIMH (Messina et al., 2021) [ST1: 5th, ST3: 4th]** used transformer-based models and proposed visual-textual transformers to mainly address subtask 3 (ST3). For the visual part, they used ResNet50, and for the textual part, they used BERT. The same network used the multi-label classification on text (ST1) by using only the textual part of the network.

**Alpha (Feng et al., 2021) [ST1:2nd, ST3:1st]** team pre-trained a transformer using text with visual features. They extract grid features, using ResNet50, and salient region features, using BUTD. They used grid features to capture the high-level semantic information found in the images. Additionally, they used salient region features to describe objects and to caption the event present in the memes. For ST1, they combined the text and the text representation of the visual features, and trained DeBERTa. For ST3, they built an ensemble of fine-tuned DeBERTa+ResNet, DeBERTa+BUTD, and ERNIE-VIL.

**HOMADOS (Kaczyński and Przybyła, 2021) [ST2: 2nd]** used a multi-task learning (MTL) approach with additional datasets such as the PTC corpus from SemEval-2020 (Da San Martino et al., 2020a), and a fake news corpus (Przybyła, 2020). The model was trained using BERT followed by several output layers, which solve auxiliary tasks of propaganda detection and credibility assessment in two distinct scenarios: sequential and parallel MTL, effectively accelerating the training process. The final submission used a parallel MTL approach on the propaganda detection of SemEval-2020, which ranked second.

**TeamFPAI (Xiaolong et al., 2021) (ST2: 3rd)** formulated the task as a question answering one in a machine reading comprehension (MRC) framework, which achieved better results compared to an ensemble-based approach (Liu et al., 2018). Moreover, data augmentation and loss design techniques were also explored to alleviate the problem of data sparseness and imbalance. Their system was ranked 3rd in the final evaluation phase.

**CSECUDSG (Hossain et al., 2021) (ST1: 13th, ST2: 6th, ST3: 6th)** participated in all three subtasks. For ST1, they used a majority vote late fusion on top of logistic regression, decision tree, and fine-tuned DistilBERT models. For ST2, they reformulated the task as one of multi-label classification, where a pre-trained BERT model was used to design binary classifiers for each technique in a multi-label classification setting. For ST3, they used a majority voting late fusion on top of fine-tuned DistilBERT, ResNet50, and a predicted label from an early fusion model. The early fusion model consisted of features from (i) multi-kernel CNN on top of the LSTM model with word embeddings including (ii) word2vec (Mikolov et al., 2013), (iii) word embeddings fine-tuned FastBERT (Liu et al., 2020), (iv) RoBERTa, (v) sentence embeddings from FastBERT, (vi) image features from YouTube-8M (Abu-El-Hajja et al., 2016), and (vii) multimodal features from VisualBERT (Li et al., 2019).

**LeCun (Dia et al., 2021) [ST1: 6th]** trained five models and combined them in an ensemble. Initially, they pre-processed text using stemming. Later, they trained DeBERTa and RoBERTa models with augmented data using synonym replacement, random insertion, random swap, random deletion and back-translation. They first trained the five models separately, and then they fine-tuned the ensemble on the official non-augmented data.

**LIIR (Ghadery et al., 2021)[ST3: 8th]** used data augmentation through back-translation and CLIP to obtain image and text representations, which were then fed to a chained classifier that uses the correlations between the output techniques.

**LT3-UGent (Singh and Lefever, 2021) [ST3: 14th]** participated in subtask 3 only. They used Multimodal Compact Bilinear Pooling to combine representations from ResNet-51 and BERT. They further fine-tuned on the PTC corpus (Da San Martino et al., 2020a).

**MinD (Tian et al., 2021) [ST1: 1st, ST3: 2nd]** used five pre-trained models for ST1: BERT, RoBERTa, XLNet, DeBERTa, and ALBERT. They first fine-tuned them on the PTC corpus (Da San Martino et al., 2020a), and then on the training data. For the final prediction, they averaged the probabilities of the models. They also used a post-processing rule: a bigram that appeared more than three times was flagged as a *Repetition*. The system for ST1 was also used for ST3, combined with (i) ResNet-34, a face recognition system, (ii) OCR-based positional embeddings for text boxes in the image, and (iii) Faster R-CNN to extract region-based image features. They combined the textual and the visual representations by averaging their probabilities. Other multimodal fusion strategies included concatenation of the representation and mapping them to the space using a multilayer perceptron.

**NLP-IITR (Gupta and Sharma, 2021) [ST1: 15th]** used an ensemble that included included fine-tuned RoBERTa, BERT, and three additional models. They further used pre-processing. To tackle data scarceness for some rare labels, they used data augmentation using back-translation.

**NLyticsFKIE (Pritzkau, 2021) [ST1: 9th, ST3: 13th]** used RoBERTa as a text encoder in ST1 and ST3. For ST1, they used RoBERTa’s output to build a classifier to predict each label separately. For ST3, they still used RoBERTa to encode the text and a VGG-16 layer to encode the image. They used multiple copies of a cross-modality encoder that outputs an encoding of the image features with respect to the text features, and vice versa. The concatenation of the two cross-encoders’ outputs was then passed through a residual layer followed by layer normalization.

**Volta (Gupta et al., 2021) [ST1: 3rd, ST2: 1st, ST3: 5th]** used a combination of transformers for all subtasks. For ST1, they used RoBERTa’s [CLS] token, which they fed to a feed-forward neural network, and example weighting to take care of class imbalance. For ST2, they predicted token classes by considering the output of each token embedding as obtained by RoBERTa. To account for subwords’ class, they merged each subword belonging to the same token and assigned the union of the subwords’ labels. For ST3, they separately encoded the textual features (extracted using RoBERTa) and the multi-modal features (extracted using UNITER,

VisualBERT, and LXMERT). This layer’s input was a sequence of textual subwords and visual tokens extracted by keeping the top 36 regions of interest as returned by Faster R-CNN. A concatenation of the two different [CLS] tokens was then fed into an MLP, and weighted labels were used with a cross-entropy loss.

**WVOQ (Roele, 2021) [ST2: 5th]** used a novel approach to ST2 consisting of adopting an encoder–decoder strategy. The encoder encodes the passage, while the decoder generates a marked version of the input, where the markup outlines the various spans along with the classes they belong to. In this way, the system performed simultaneous span detection and classification. The encoder–decoder used a specialization of BART.

**YNU-HPCC (Zhu et al., 2021) [ST1: 12th, ST2: 7th, ST3: 11th]** For ST1, they used a CNN on top of ALBERT and fine-tuned the model for multi-label classification. For ST2, each propaganda technique was considered as an independent task, and features were extracted from the pre-trained BERT model. Subsequently, the problem was addressed as a multi-task sequence labeling one, and the results for each task were combined. For ST3, a multi-modal network was used, where embeddings from textual and visual networks were concatenated, which was followed by a fully connected layer. For the text, the same approach was used for ST1, and for the image, ResNet and VGGNet were used for image feature extraction.

# Alpha at SemEval-2021 Task 6: Transformer Based Propaganda Classification

Zhida Feng<sup>1,2</sup>\*, Jiji Tang<sup>1</sup>\*, Jiayang Liu<sup>1</sup>, Weichong Yin<sup>1</sup>, Shikun Feng<sup>1</sup>, Yu Sun<sup>1</sup>, Li Chen<sup>2</sup>

<sup>1</sup>Baidu Inc., Beijing, China

<sup>2</sup>Wuhan University of Science and Technology, China

{fengzhida, tangjiji, liujiayang, yinweichong, fengshikun01, sunyu02}@baidu.com  
chenli@wust.edu.cn

## Abstract

This paper describes our system participated in Task 6 of SemEval-2021: this task focuses on multimodal propaganda technique classification and it aims to classify given image and text into 22 classes. In this paper, we propose to use transformer-based (Vaswani et al., 2017) architecture to fuse the clues from both image and text. We explore two branches of techniques including fine-tuning the text pre-trained transformer with extended visual features and fine-tuning the multimodal pre-trained transformers. For the visual features, we experiment with both grid features extracted from ResNet(He et al., 2016) network and salient region features from a pre-trained object detector. Among the pre-trained multimodal transformers, we choose ERNIE-ViL (Yu et al., 2020), a two-stream cross-attended transformers model pre-trained on large-scale image-caption aligned data. Fine-tuning ERNIE-ViL for our task produces a better performance due to general joint multimodal representation for text and image learned by ERNIE-ViL. Besides, as the distribution of the classification labels is extremely unbalanced, we also make a further attempt on the loss function and the experiment results show that focal loss would perform better than cross-entropy loss. Lastly, we ranked first place at sub-task C in the final competition.

## 1 Introduction

Propaganda is usually adopted to influence the audience by selectively displaying the facts to encourage specific synthesis or perception, or using the loaded language to produce emotion rather than emotion itself. It was often associated with materials prepared by governments in the past century. In the internet era, activist groups, companies, religious organizations, the media, and individuals also

produce propaganda, and sometimes it can reach very large audiences (Da San Martino et al., 2020). With the recent research interest in detecting “fake news”, the detection of persuasion techniques in the texts and images has emerged as an active research area. Most previous work like (Patil et al., 2020) and (Chauhan and Diddee, 2020) have performed the analysis at the language content level only. However, in our daily life, memes consist of images superimposed with texts. The aim of the image in a meme is either to reinforce a technique in the text or to convey one or more persuasion techniques.

SemEval-2021 Task6-c offers a different perspective, multimodal multi-label classification (Dimitrov et al., 2021), identify which of the 22 techniques are used both in the textual and visual content of memes. Since memes are combinations of texts and images, for this propaganda classification task, we proposed to use transformer-based architecture to fuse the clues from both linguistic and visual modalities. Two branches of fine-tuning techniques are explored in this paper. First, a text pre-trained transformer is applied with extended visual features. Specifically, we initialize the transformer with pre-trained text transformers and fine-tune the model with extended visual features including grid features(e.g., ResNet(He et al., 2016)) and region features(e.g., BUTD (Anderson et al., 2018)) from an image feature extraction network and an object detector respectively. Second, pre-trained multimodal transformers from ERNIE-ViL(Yu et al., 2020) are used due to its better multimodal joint representations characterizing cross-modal alignments of detailed semantics.

Our contributions are three-folds:

- We propose to use transformer architecture for fusing the visual and linguistic clues to tackle the propaganda classification task.

\*indicates equal contribution.

- We find that the multimodal pre-trained transformers work better than using text pre-trained transformers with visual features. And the experiment results have shown that fine-tuning the ERNIE-ViL model could achieve state-of-the-art performance for this task.
- Our ensemble result of several models obtains the best score and ranks first in Semeval-2021 Task 6-c multimodal classification task.

## 2 Related work

### 2.1 Text Transformers

Transformer network (Vaswani et al., 2017) is first introduced in neural machine translation in which encoder and decoder are composed of multi-layer transformers. After then, pre-trained language models, such as BERT (Devlin et al., 2018) and GPT (Radford et al., 2018), adopting transformer encoder as the backbone network, have significantly improved the performance on many NLP tasks. One of the main keys to their success is the usage of transformer to capture the contextual information for each token in the text via self-attention. Later text pre-training works, such as ERNIE2.0 (Sun et al., 2020), RoBERTa (Liu et al., 2019) and XLNET (Yang et al., 2019) are all shared the same multi-layer transformer encoder and mainly put their effort on modification of pre-training task.

### 2.2 Visual Feature Extraction

Visual feature extractors are mainly composed of plenty of convolutional neural networks (CNN) since CNN has a strong ability to extract complex features that express the image with much more details and learn the task-specific features much more efficiently. Existing works can be divided into the following two types which are based on two different image inputs: image grids and object regions. Some of those methods, such as VGG (Simonyan and Zisserman, 2014), ResNet (He et al., 2016) operate attention on CNN features corresponding to a uniform grid of equally-sized image regions. While the other works like Faster R-CNN (Ren et al., 2015) operate a two-stage framework, which firstly identifies the image regions containing the specific objects, and then encodes them with multi-layer CNNs.

### 2.3 Multimodal Transformers

Inspired by text pre-training models (Devlin et al., 2018), many cross-modal pre-training models for

vision-language have been proposed. To integrate visual features and text features, recent multimodal pre-training works are mainly based on two variables of transformers. Some of them, like UNITER (Chen et al., 2019) and VILLA (Gan et al., 2020) use a uniform cross-modal transformer modelling both image and text representations. As fine-tuning on multimodal classification tasks, such as the Visual-question-answering (VQA) (Antol et al., 2015) task (a multi-label classification task), unified transformers take textual and visual features as the model input, treat the final hidden state of  $h_{[CLS]}$  as the vision-language feature. While the others like Vilbert (Lu et al., 2019), LXMERT (Tan and Bansal, 2019), ERNIE-ViL (Yu et al., 2020) are based on two-stream cross-modal transformers, which bring more specific representations for image and text. These two transformers are applied to images and texts to model visual and textual features independently and then fused by a third transformer in a later stage. The fusion of the final hidden state of  $h_{[CLS]}$  and  $h_{[IMG]}$  are used to do the classification.

## 3 Approach

We propose to use a transformer encoder to fuse the clues from both linguistic and visual modalities and our approach is summarized in two branches, the first one is fine-tuning a text pre-trained transformer with extended visual features, and the other one is fine-tuning a multimodal pre-trained model. For the first one, we try two different sets of visual features, grid features based on equally-split patches of the image and salient region features based on an object detector. For the second one, a SoTA multimodal model, ERNIE-ViL (Yu et al., 2020) is applied with a multi-label classification loss. A unified framework for the two branches is shown in Figure 1. We will introduce more details in this section.

### 3.1 Text Pre-trained Transformer with Visual Features

Our model consists of three parts: a) input feature extractor, b) feature fusion encoder, c) classification encoder.

For the first part, the text is tokenized into subwords to lookup the embedding while the image is processed by a feature extractor, such as a grid feature processor or a salient region feature processor to convert into vision embeddings. The input em-



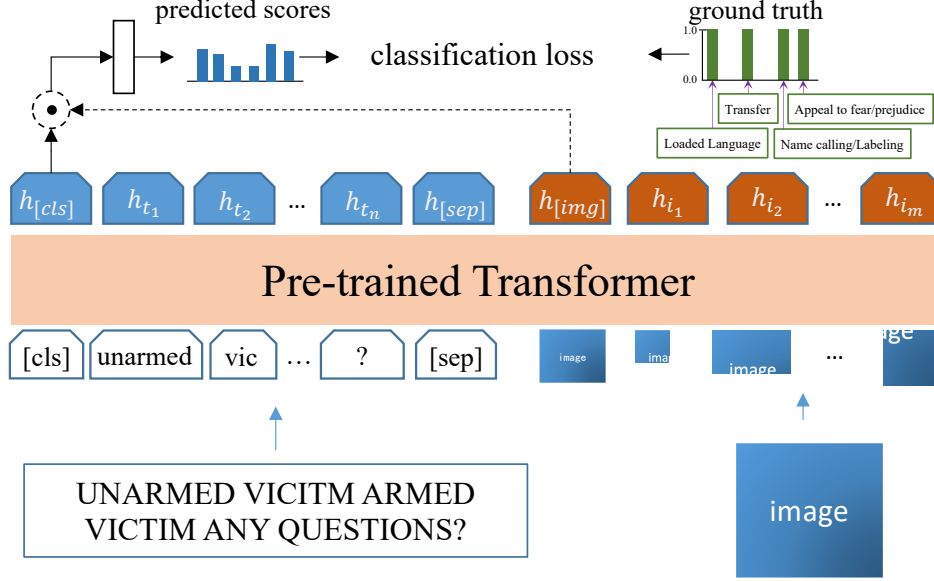


Figure 1: A unified framework used for the multimodal classification task.

beddings are combinations of image embeddings and text embeddings and represented as

$$h_{[CLS]}, h_{t_1}, \dots, h_{t_n}, h_{[SEP]}, h_{i_1}, \dots, h_{i_m}, h_{[SEP]}$$

where the  $h_{[CLS]}$ ,  $h_{[SEP]}$  are the vector representations of special tokens  $[CLS]$  and  $[SEP]$  respectively. The  $[CLS]$  token is inserted in the beginning of the sequence, which act as an indicator of the whole text, specifically, it is used to perform complete text classification. The  $[SEP]$  is a token to separate a sequence from the subsequent one and indicate the end of a text.  $h_{t_1}, \dots, h_{t_n}$  are the text embeddings, and  $h_{i_1}, \dots, h_{i_m}$  are the vision embeddings. For the vision embeddings part, grid features and salient region features are used.

**Grid Features** Convolutional neural networks have potent capabilities in image feature extraction. The feature map obtained after the image goes through multiple stacked convolution layers contains high-level semantic information. Given an image, we can use a pre-trained CNN encoder, such as ResNet, to transform it to a high-dimensional feature map and flatten each pixel on this feature map to form the final image representation.

**Salient Region Features** Object detection models are widely used to extract salient image regions from the visual scene. Given an image, we use a pre-trained object detector to detect the image regions. The pooling features before the multi-class

classification layer are utilized as the region features. The location information for each region is encoded via a 5-dimension vector representing the fraction of image area covered and the normalized coordinates of the region and then is projected and summed with the region features.

For the second part, the transformer encoder fuses the input text and image embedding, and finally a cross-modal representation of size  $D$  is achieved for this sequence.

The last part of our model is the classification encoder and loss function. After obtaining the encoding representation of the image and the text from the transformer encoder, we send the representation of  $[CLS]$  through the classification head, which is consisted of a fully connected layer and a *Sigmoid* activation for predicting the score of each category and loss with the ground truth.

### 3.2 Multimodal Pre-trained Transformer

Different from a single-modal pre-trained text transformer described above, a multimodal pre-trained transformer for vision-language can learn more efficient presentations. In this part, a SoTA model, ERNIE-ViL, is applied.

For the generation of input embedding of text and image, it is mostly the same as the procedure described in the previous section. Differences are two-folds. First, for the vision feature, a faster R-CNN encoder (Anderson et al., 2018) is used to detect the salient regions while the position infor-

mation is taken into consideration. Second, The text and the visual input embedding is represented as

$$h_{[CLS]}, h_{t_1}, \dots, h_{[SEP]}, h_{[IMG]}, h_{i_1}, \dots, h_{i_m}$$

where there is a new token  $h_{[IMG]}$  represents the feature for the entire image.

For the feature fusion part, ERNIE-ViL utilized a two steam cross-modal transformer to fuse the multimodal information. For more details, you may refer to (Yu et al., 2020).

### 3.3 Criterion

In this task, there are 22 classes and the distribution of positive and negative samples is extremely unbalanced. To solve this problem, we use the focal loss to improve the imbalance of positive and negative samples. For  $i$ -th class

$$L_{class_i} = \begin{cases} \alpha(1-p)^\gamma \log(p) & \text{if } y=1 \\ (1-\alpha)p^\gamma \log(1-p) & \text{otherwise} \end{cases}$$

where  $y$  is the ground truth;  $p$  is model prediction, which is the confidence score of category  $i$ ;  $\alpha$  and  $\gamma$  are hyper-parameters,  $\alpha$  is used to control the loss weight of positive and negative samples, and  $\gamma$  is used to scale the loss of difficult and easy samples.

## 4 Experiment

### 4.1 Implementation Details

In this task, we choose DeBERTa-large+ResNet50, DeBERTa-large+BUTD and ERNIE-ViL as the final models. We performed all our experiments on a Nvidia Tesla V100 GPU with 32 GB of memory. The models are trained for 20 epochs and we pick the model which has the best performance on validation set.

For the DeBERTa transformer, the Adam optimizer with a learning rate of 3e-5 is used. Also, we have applied the linear warm strategy for the learning rate. We set  $\alpha = 0.9$  and  $\gamma = 2.0$  for the focal loss. To ensure robustness under a small dataset, we set the threshold to 0.5 instead of performing a threshold search strategy on the validation set. For the pre-trained object detector, we choose Faster R-CNN (Anderson et al., 2018) and name the region features as BUTD in the experimental results.

For the ERNIE-ViL transformers, we use the same input preprocessing methods as (Yu et al.,

	Positive(%)	Negative(%)
train	1745(11.55%)	13369(88.45%)
dev	183(13.20%)	1203(86.80%)
test	523(13.49%)	3877(86.51%)

Table 1: Statistics of the positive and negative distribution of the dataset.

Loss Function	Precision	Recall	F1
cross-entropy	76.12	55.74	64.35
focal loss	71.18	66.12	68.56

Table 2: Results of different loss functions.

2020) and choose the large scale model<sup>1</sup> pre-trained on all the four datasets. We finetune on our multimodal classification dataset with a batch of 4 and a learning rate of 3e-5 for 20 epochs.

### 4.2 Experimental Analysis

#### 4.2.1 DeBERTa with Visual Features

**Unbalanced Distribution** There are 687/63/200 examples includes 22 categories in the train/validation/test datasets respectively. As shown in Table 1, the distribution of the classes is extremely unbalanced. If the cross-entropy loss is adopted directly during model training(the visual features are from ResNet50), the model output may have a greater chance of predicting the majority class(negative class in this task), which results in a lower recall. To solve this problem, the focal loss is applied. From Table 2, it can be seen that the result with focal loss performs much better than with cross-entropy loss respective to the F1 score.

**Visual Features** We evaluate the improvement brought by extended visual features and explore different types of visual feature extractors, e.g., from pre-trained image classification networks or pre-trained object detectors. The results are illustrated in Table 3. Firstly, it can be seen that the final score is significantly improved with mixing image features compared with using only text features (Row “w/o vision feature”), which indicates that the visual information is significantly beneficial for recognizing cross-modal propaganda techniques. Then, for features extracted from ResNet, we find that the depth of the network affects the results, especially on the validation dataset, with the best result from ResNet50. The reason may be

<sup>1</sup>the pre-trained model is downloaded from <https://github.com/PaddlePaddle/ERNIE/tree/repro/ernie-vil>

	<b>dev-F1</b>	<b>test-F1</b>
w/o vision feature	65.73	55.10
ResNet18	65.92	55.59
ResNet50	68.56	55.96
ResNet152	65.91	55.63
BUTD	66.29	56.21

Table 3: The results of using features extracted different networks.

<b>region numbers</b>	<b>Dev F1</b>	<b>Test F1</b>
5	64.91	54.00
10	66.67	54.60
36	67.40	57.14
100	67.45	56.07

Table 4: Results comparisons with different object region number inputs.

that the shallower network has insufficient feature extraction capabilities, and the deeper network is very difficult to train. Finally, the region features from the pre-trained object detector (Row “BUTD”) work best with an improvement of 0.25 on the test dataset compared to ResNet50 features.

#### 4.2.2 ERNIE-ViL

We compare the performance between ERNIE-ViL with different object region inputs, which are number dynamic ranges between 0 and 36 with a fixed confidence threshold of 0.2 and constantly fixed 5, 10, or 100 boxes. The results are illustrated in Table 4.

Results show that a larger box number can always achieve better performance within a certain range. Utilizing 0-36 boxes leads to huge performance improvement with a 3.14 and 2.54 on Test-F1 compared with using constant 5 boxes and constant 10 boxes respectively. It can be concluded that more object regions in a certain range can provide more useful information. However, the performance with 100 boxes is worse than that with 0-36 boxes. The reason may lie in that there are not enough objects in the task sample. The ex-

<b>Models</b>	<b>Dev-F1</b>	<b>Test-F1</b>
DeBERTa + ResNet50	68.56	55.96
DeBERTa + BUTD	66.29	56.21
ERNIE-ViL	67.40	57.14
Ensemble	69.12	58.11

Table 5: Final ensemble result.

tracted low-confidence object regions may mislead the multimodal model, therefore fuse useless or harmful visual features with text features. As a result of that, brings a performance decrease on the final score.

#### 4.3 Ensemble Results

The performance comparison between our two branches of approach is shown in Table 5. It can be concluded that fine-tuning the multimodal pre-trained transformer (Row “ERNIE-ViL”) works better than fine-tuning text pre-trained transformers with visual features (Row “DeBERTa + BUTD”). Overall, fine-tuning ERNIE-ViL has achieved state-of-the-art performance for this multimodal classification task.

Since the training dataset is small, we train multiple models under various model structures and different parameter configurations to take full advantage of the training dataset and increase the diversity of models. We choose three models of all model structures and all parameter configuration that performs best on the validation set and then ensemble them together. After performing ensemble strategy on those three models, both validation and test scores increases. As a result of that, we achieved a 58.11 score at F1 in the test set and ranked first place in the task competition.

### 5 Conclusion

We explore two branches to fine-tune pre-trained transformers to jointly modelling texts and images for the propaganda classification task. The first branch, fine-tuning pre-trained text transformer with visual feature, obtain significant performance improvement compared to text classification which validate the importance of visual clues for this task. Visual features from object detector yield slightly better results than grid features from ResNet. Importantly, fine-tuning pre-trained multimodal transformers obtain the best single model performance. And this improvement further validates the claim made by previous work that vision-language pre-training learned general joint representation needed for multimodal tasks. Besides, since the distribution of the classification labels is extremely unbalanced, we also make a further attempt on the loss function. Training models with focal loss can lead to a huge performance improvements than training with cross entropy loss.

## References

- Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.
- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433.
- Aniruddha Chauhan and Harshita Diddee. 2020. Psuedoprop at semeval-2020 task 11: Propaganda span detection using bert-crf and ensemble sentence level classifier. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1779–1785.
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Uniter: Learning universal image-text representations.
- Giovanni Da San Martino, Alberto Barrón-Cedeno, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. Semeval-2020 task 11: Detection of propaganda techniques in news articles. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval '21, Bangkok, Thailand*.
- Zhe Gan, Yen-Chun Chen, Linjie Li, Chen Zhu, Yu Cheng, and Jingjing Liu. 2020. Large-scale adversarial training for vision-and-language representation learning. *arXiv preprint arXiv:2006.06195*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*.
- Rajaswa Patil, Somesh Singh, and Swati Agarwal. 2020. Bpgc at semeval-2020 task 11: Propaganda detection in news articles with multi-granularity knowledge sharing and linguistic features based ensemble learning. *arXiv preprint arXiv:2006.00593*.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*.
- Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.
- Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Fei Yu, Jiji Tang, Weichong Yin, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. 2020. [Ernie-vil: Knowledge enhanced vision-language representations through scene graph](#).

# SemEval-2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense

J.A. Meaney<sup>1</sup>, Steven R. Wilson<sup>1</sup>, Luis Chiruzzo<sup>2</sup>, Adam Lopez<sup>1,3</sup>, Walid Magdy<sup>1,4</sup>

<sup>1</sup> School of Informatics, The University of Edinburgh, Edinburgh, UK

<sup>2</sup> Universidad de la República, Uruguay

<sup>3</sup> Rasa

<sup>4</sup> The Alan Turing Institute, London, UK

{jameaney, steven.wilson}@ed.ac.uk

{alopez, wmagdy}@inf.ed.ac.uk

luischir@fing.edu.uy

## Abstract

SemEval 2021 Task 7, HaHackathon, was the first shared task to combine the previously separate domains of humor detection and offense detection. We collected 10,000 texts from Twitter and the Kaggle Short Jokes dataset, and had each annotated for humor and offense by 20 annotators aged 18-70. Our subtasks were binary humor detection, prediction of humor and offense ratings, and a novel controversy task: to predict if the variance in the humor ratings was higher than a specific threshold. The subtasks attracted 36-58 submissions, with most of the participants choosing to use pre-trained language models. Many of the highest performing teams also implemented additional optimization techniques, including task-adaptive training and adversarial training. The results suggest that the participating systems are well suited to humor detection, but that humor controversy is a more challenging task. We discuss which models excel in this task, which auxiliary techniques boost their performance, and analyze the errors which were not captured by the best systems.

## 1 Introduction

Humor is a key component of many forms of communication, and so it is commanding an increasing amount of attention in the natural language processing (NLP) community (Attardo, 2008; Taylor and Attardo, 2017; Amin and Burghardt, 2020). However, like much of figurative language processing, humor detection requires a different perspective on several traditional NLP tasks. For example, the problem of reducing lexical or syntactic ambiguity differs when ambiguity is key to some humor mechanisms. Tackling these challenges has the potential to improve many downstream applications, such as content moderation and human-computer interaction (Rayz, 2017).

However, humor is a subjective phenomenon, which evokes varying degrees of funniness in its audience, while also provoking other reactions such as offense, in certain listeners. The perception of humor is known to vary along the lines of age, gender, personality and other factors (Ruch, 2010; Kuipers, 2015; Hofmann et al., 2020). That humor can also evoke offense may be partly due to differences in acceptability judgements across demographic groups, and may also be in part due the use of humor to mask hateful or offensive content (Sue and Golash-Boza, 2013). Lockyer and Pickering (2005) expand on this by highlighting that it is common for societies to explore the link between humor and offense, free speech and respect.

HaHackathon is the first shared task to combine humor and offense detection, based on ratings from a wide variety of demographic groups. Task participants were asked to detect if a text was humorous and to predict its average ratings for both humor and offense. We also introduce a novel humor controversy detection task, which represents the extent to which annotators agreed/disagreed with each other over the humor rating of a joke. A humorous text was labelled as controversial if the variance in the humor ratings was higher than the median humor rating variance in the training set.

## 2 Related Work

Computational humor detection is a relatively established area of research. Taylor and Mazlack (2004) were one of the first to explore recognising wordplay with ngrams. Mihalcea and Strapparava (2005; 2006) experimented with 16,000 one-liners and 16,000 non-humorous texts, using a feature-driven approach. More recently, Zhang and Liu (2014) turned to online domains, by detecting humor on Twitter with a view to improving downstream tasks such as sentiment analysis and opinion

mining.

Workshops on humor detection have become more prominent with each shared task, and have attracted many new researchers to the field. SemEval 2017 (Potash et al., 2017) featured Hashtag Wars, a humor task with a unique data annotation procedure. This task featured tweets that had been submitted in response to a number of comedic hashtags released by a Comedy Central program. The top-10 response tweets were selected by the show’s producers and the winning tweet was selected by the show’s audience. Based on these labels, (top-10, winning tweet, and other) the sub-tasks required competitors to predict the labels, and to predict which text was funnier, given a pair tweets. The winning systems were split between feature-driven support vector machines (SVMs) and recurrent neural networks (RNNs).

The first Spanish-language humor detection challenges were the HAHA tasks in 2018 (Castro et al., 2018) and 2019 (Chiruzzo et al., 2019). These collected data from more than fifty different humorous Twitter accounts, representing a wide variety of humor genres. The sub-tasks asked competitors to predict if a text was humorous, and to predict the average funniness score given to the humorous texts. In the first year, the top teams used evolutionary algorithms to optimize linear models like Naive Bayes, as well as bi-directional RNNs. In the second year, the top teams started to use pre-trained language models (PLMs) like BERT (Devlin et al., 2018) and ULMFit (Howard and Ruder, 2018).

Most recently, Hossain et al. (2020) generated data for their task by collecting news headlines, and asking annotators to make a micro-edit to the headline to render it funny. These edited headlines were rated for funniness by other annotators. The sub-tasks were to rank the funnier of two edits, and to predict the average funniness score given by the annotators. The winning teams used ensembles of various PLMs, and RNNs.

### 3 Data

#### 3.1 Data Collection

In order to examine naturally-occurring humorous and offensive content in English, we sourced 80% of our data from Twitter. The remaining 20% of texts, we selected from the Kaggle Short Jokes dataset<sup>1</sup> for the following reasons:

<sup>1</sup><https://www.kaggle.com/abhinavmoudgil195/short-jokes>

Target	Keywords
Sexism	She, woman, mother, girl, b*tch, he, man, blond, p*ssy, hooker, slut, wh*re
Body	Fat, thin, skinny, tall, short, bald, amputee, redneck
Origin	Mexico, Mexican, Ireland, Irish, Indian, Pakistan, China, Chinese, Polish, German, France, Welsh, Vietnam, Asian, American, Russia, Arab, Jamaican, homeless
Sexual Orientation	Gay, lesbian, d*ke, f*ggot, homo, aids, LGBT, trans, tr*nny
Racism	Black, Africa, African, wop, n***** white people,
Ideology	Feminism, leftie/lefty
Religion	Muslim, Islam, Jew, Jewish, Catholic, Protestant, Hindu, Buddhist, ISIS, Jesus, Mohammed
Health	Wheelchair, blind, deaf, r*tard, Steven Hawking, Stevie Wonder, Helen Keller, dyslexic

Table 1: Targets and Sample Keywords

- **Humor Quota:** To ensure that a sample of texts in the dataset were intended to be humorous. Our annotation procedure asks raters if the intention of the text is to be humorous (as evidenced by the the setup/punchline structure, or absurd content). As the texts were sourced from the /r/jokes and /r/cleanjokes subreddits, we were confident that the intention of the text was to be humorous.
- **Traditional Humor Quota:** We wanted to represent jokes which have a traditional setup and punchline structure. Twitter humor is known to use a number of unique features (Zhang and Liu, 2014), which may not be equally recognisable to all annotators and so we wanted to have a selection of conventionally recognisable texts in order to gauge what the audience response was, and to use as a quality check for annotators (see below).
- **Offense Quota:** To ensure that a proportion of texts were likely to be considered offensive by the annotators, half of the texts selected according to the procedure below.

To select potentially offensive texts, we used some of the keywords associated with Silva et al.’s (2016) sub-categories of hate speech in social media, and queried the Kaggle dataset for these.

Text	Keyword = Target
A <b>fat</b> woman just served me at McDonalds and said "Sorry about the wait". I replied and said, "Don't worry, you'll lose it eventually".	Yes
Don't worry if a <b>fat</b> guy comes to kidnap you... I told Santa all I want for Christmas is you.	No

Table 2: Sample of potentially offensive and non-offensive texts

From these texts, we identified the target, or butt, of the joke and made the assumption that a text could be potentially offensive to our annotators if the hate speech keyword was the target of the joke. We selected 1,000 texts this way. We also assumed that the text would likely be considered not offensive if the keyword was mentioned, but was not the target and selected a further 1,000 texts like this. This was to reduce the probability that a humor/offense detection system would learn to classify texts simply based on the presence of a hate speech keyword.

### 3.1.1 Selection of Twitter texts

In order to avoid introducing annotation confounds such as a lack of cultural or linguistic knowledge (Meaney, 2020), we selected the texts and the annotators from the same region – the US. When sourcing the humorous Twitter data, we selected accounts according to whether they were based in the US and posted almost exclusively humorous content (e.g. @humorous1liners, @conanobrien). For the non-humorous Twitter accounts, we elected not to use news sources, e.g. CNN due to stylistic differences between news and humor (Mihalcea and Strapparava, 2006) making them easy to differentiate. The non-humorous accounts we selected centred on US celebrities (e.g. @thatonequeen, @Oprah), organisations that represent the targets of hate speech groups (e.g. @BlkMentalHealth, in order to increase the occurrences of the keywords in a non-humorous and non-offensive context), trivia accounts (e.g. @UberFacts, as the question and answer structure is similar to some types of setup and punchline) and tv/movie quotation accounts (e.g. @MovieQuotesPage, in order to resemble the dialogue-type jokes that are common on Twitter). Please see the appendix for a comprehensive list of accounts.

Using the Twitter API, we crawled up to 2,000 tweets from each account, and removed retweets and texts containing links. We also removed tweets that contained references to US Politics, the pandemic, or TV show characters as topical humor can

be difficult to understand once the event it is tied to has passed (Highfield, 2015). From an initial 76,542 texts, we were left with 8,000 tweets. From these, we removed hashtags that labelled the texts as humorous, e.g. #joke, and using Ekphrasis (Baziotis et al., 2017) we split up any remaining hashtags into their constituent words so as to make them less easy to differentiate from the Kaggle texts.

### 3.2 Annotation

We recruited annotators from the Prolific<sup>2</sup> platform. Participants were recruited based on their self-reported native English-speaker status, US citizenship, and membership of one of the following age groups: 18-25, 26-40, 41-55, 56-70. Each text was annotated by 5 members of each age group, giving a total of 20 annotations per text. Batches comprised 100 texts, and annotators answered the following questions:

1. Is the intention of this text to be humorous?
2. Is this text generally offensive?
3. Is this text personally offensive?

In the case that a user answered ‘yes’ to any of these questions, they were asked to rate the humor or offense from 1-5 (see figure 1). For the humor rating, the user was also given the option to select ‘I don’t get it’, meaning that they recognised by the structure or content that the text was intended to be humorous, but that they were unsure of why the text was funny. This is distinct from a rating of 1, which is a recognition of humor, with little appreciation for it.

The annotator instructions outlined that the first annotation question was intended to determine the *genre* of the text, and should be distinguished from *funniness*. Annotators were instructed to look at the structure of the joke, e.g. setup and punchline, or the content of the joke, e.g. absurdity, in order to determine if the intention was to be humorous.

<sup>2</sup><https://www.prolific.co/>

In terms of offense, we posed two annotation questions in order to avoid ambiguity about which type of offense was meant. We instructed annotators to consider as generally offensive, a text which targets a person or group of people, simply for belonging to a certain group. Alternatively, they could select yes for generally offensive if they thought that a large number of people were likely to be offended by the joke. The last question asked annotators if they felt personally offended by the text, or if they felt offended on another person’s behalf. We used only the generally offensive ratings in this task.

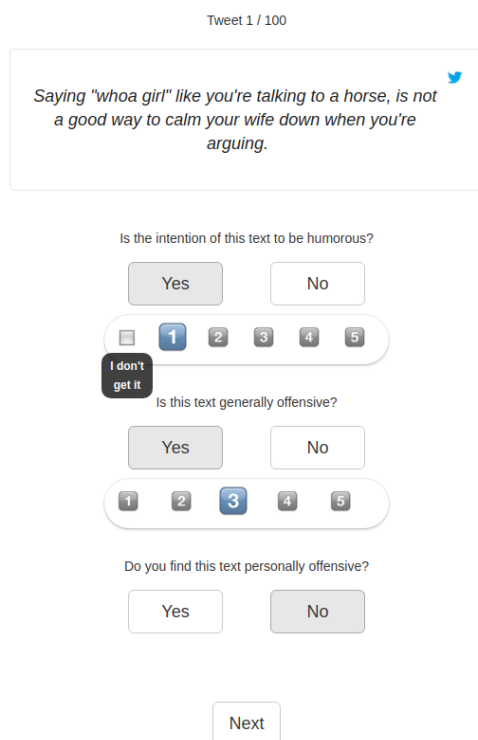


Figure 1: Screenshot from the tool used to annotate the texts.

### 3.3 Quality Control and Data Discarded

Each batch of 100 texts comprised approximately 20% of texts from Kaggle. As the majority of these have a setup and punchline structure, or other recognisable humor traits, we used these as a quality control. If an annotator did not label at least 60% of these as humor, it was clear that they they did not follow the instructions for the first question, and annotated based on perceived humor, as opposed to observation of humorous characteristics. We therefore discarded these submissions and replaced the annotators. Of 2,364 annotation sessions (e.g.

batches of 100), 301 submissions were discarded and replaced, and the ratings of the remaining 2,062 annotation sessions make up the dataset. Of these, 1,569 annotators rated one batch of texts with an additional 492 doing a second batch.

### 3.4 Data Statistics

Post-annotation, we classed a text as humorous if the majority of its twenty votes labelled it as such. In a small number of cases where votes were tied, we assigned the label humorous. For the texts labelled humorous, we calculated the average humor score, which was the average of the numerical votes. “No” ratings did not count towards this value, and votes of “I don’t know” were counted as 0, because this was deemed to be a recognizable humor structure, but one in which the humor was not successful.

Label	Affirmative	Negative	Average Rating
Humorous	6179	3821	2.24
Controversial	3052	3017	N/A
Offensive	5754	4246	1.02

Table 3: Data Statistics

The humor controversy label was based on whether the variance between the humor ratings was higher or lower than the median variance in the training set (median  $s^2 = 1.79$ ). The offense rating was the average of all ratings given, including ‘no’ as 0. Table 3 summarises the labels in the dataset, and in the case of offense, affirmative indicates that the rating is higher than 0.

Ratings	Krippendorff’s $\alpha$
Class label	0.736
Humor rating	0.124
Offense rating	0.518

Table 4: Inter-annotator agreement (Krippendorff’s  $\alpha$ ) for ratings used in subtask 1a, 1b and 2

The dataset was split 80:10:10 for training, development and test sets. The texts and annotations will continue to be available on the Codalab website, and the tweet ids, and usernames will be retained for non-commercial research use, in line with the Twitter Academic Developer Policy.

## 4 Task Description and Evaluation

We divided our tasks into four subtasks.



### Task 1a: Humor Detection

This was a binary classification task to detect, given a text, if the majority label assigned to it was humorous or not. This was evaluated using F-score for the humorous class and overall accuracy

$$Accuracy = \frac{C}{N}$$

$$F_1 = 2 * \frac{Precision \times Recall}{Precision + Recall}$$

### Task 1b: Humor Rating Prediction

This was a humor rating regression task. Participants predicted the average rating given to texts from 0-5. Texts which had not been labelled as humorous by our annotators did not have a humor rating, and predictions for these texts were not counted towards the final score by our scoring system. The metric for this task was root mean squared error (RMSE).

$$RMSE = \sqrt{\sum_{i=1}^n \left(\frac{y_i - \hat{y}_i}{N}\right)^2}$$

### Task 1c: Humor Controversy Detection

This task was also a binary classification task to predict whether the humor ratings given to the text showed it to be controversial or not. This was based on the variance in the ratings being higher or lower than the median variance in the training set humor ratings. This was also evaluated using F-score and accuracy.

### Task 2: Offense Detection

This was an offense rating regression task. Unlike the humorous task, this rating was not dependent on the text having been labelled as humorous. All annotator ratings were considered, and each text had a rating from 0-5. The metric was RMSE.

## 5 Benchmark Systems

We created simple, linear benchmarks using sklearn (Pedregosa et al., 2011) for the classification tasks which consists of a Naive Bayes classifier with bag of words features. For the regression tasks, we used a support vector regressor with term-frequency inverse document frequency features.

We also built a BERT-base classification/regression model which was run for one epoch, with a batch size of 16 and a learning rate of 5e-5, for all sub-tasks. As this system out-performed the linear benchmarks on all sub-tasks, we refer to this as the baseline in the rest of the paper.

## 6 Participant Systems

### 6.1 Overview

In total 63 teams submitted systems for the different tasks: 58 for task 1a, 50 for task 1b, 36 for task 1c and 48 for task 2. Tables 5, 6, 7 and 8 show the highest results for each task, with performance broken down by subsets of texts from the Kaggle jokes dataset and from Twitter. -\*/

Team	Acc	F1	Kaggle F1	Twitter F1
PALI	0.9820	0.9854	0.9949	0.9811
stce	0.9750	0.9797	0.9871	0.9764
DeepBlueAI	0.9600	0.9676	0.9949	0.9551
SarcasmDet	0.9600	0.9675	0.9949	0.9548
mengyuan_jiayi	0.9590	0.9667	0.9871	0.9574
stevenhuahua	0.9580	0.9666	0.9949	0.9538
zain	0.9580	0.9663	0.9949	0.9534
EndTimes	0.9570	0.9655	0.9897	0.9545
MagicPai	0.9570	0.9653	0.9897	0.9542
Meizizi	0.9570	0.9653	0.9871	0.9554
mmmm	0.9560	0.9647	0.9923	0.9523
baseline (BERT)	0.911	0.9283	0.9949	0.8978
baseline (Linear)	0.8570	0.8840	0.9792	0.8410

Table 5: Results of the top performing systems for participants of task 1a (humor detection), showing F1 and accuracy for the whole test set, and F1 for Kaggle texts only and tweets only.

### 6.2 Highest Ranking Systems

The top-ranking teams were selected based on F-score, in the case of a tie in accuracy score. The top-10 made extensive use of pre-trained language models such as BERT, ERNIE 2.0 (Sun et al., 2020), ALBERT (Lan et al., 2019), DeBERTa (He et al., 2020) or RoBERTa (Liu et al., 2019). Ensembling these models by majority voting or averaging scores proved to be a popular and useful approach.

Team	All	Kaggle	Twitter
abcbpc	0.4959	0.4544	0.5141
mmmm	0.4977	0.4554	0.5162
Humor@IITK	0.5210	0.4702	0.5430
YoungSheldon	0.5257	0.4587	0.5541
IIITH	0.5263	0.4821	0.5456
fdabek	0.5271	0.4836	0.5462
Amherst685	0.5339	0.4584	0.5656
-*/ gerarld	0.5393	0.4857	0.5625
CS-UM6P	0.5401	0.4927	0.5608
SarcasmDet	0.5446	0.5001	0.5641
baseline (BERT)	0.8000	0.4803	0.9117
baseline (SVM)	0.8609	0.7157	0.9205

Table 6: Results of the top performing systems for participants of task 1b (humor rating), showing RMSE for whole test set, for Kaggle texts only and tweets only.

Team	Acc	F1	Kaggle F1	Twitter F1
PALI	0.4943	0.6302	0.6667	0.6118
mmmm	0.4699	0.6279	0.6621	0.6109
SarcasmDet	0.4699	0.6270	0.6552	0.6130
EndTimes	0.4602	0.6261	0.6598	0.6097
DeepBlueAI	0.4650	0.6257	0.6621	0.6078
CS-UM6P	0.4537	0.6242	0.6598	0.6070
CHaines	0.4537	0.6242	0.6598	0.6070
Ferryman	0.4537	0.6242	0.6598	0.6070
IIITH	0.4537	0.6242	0.6598	0.6070
abcbpc	0.4537	0.6242	0.6598	0.6070
fdabek	0.4537	0.6233	0.6598	0.6057
YoungSheldon	0.4780	0.6210	0.6545	0.6049
Humor@IITK	0.4520	0.6209	0.6574	0.6033
RoMa	0.4732	0.6197	0.6503	0.6042
<i>baseline (BERT)</i>	<i>0.4731</i>	<i>0.6232</i>	<i>0.6574</i>	<i>0.6060</i>
<i>baseline (SVM)</i>	<i>0.4374</i>	<i>0.4624</i>	<i>0.4804</i>	<i>0.4529</i>

Table 7: Results of the top performing systems for participants of task 1c (humor controversy), showing F1 and accuracy for the whole test set, and F1 for kaggle texts only and tweets only.

Similarly, many teams experimented with single and multi-task learning setups, and multi-task models tended to be more successful across sub-tasks. Further improvements were achieved with domain adaptation strategies and adversarial training.

### 6.2.1 DeepBlueAI (Song et al., 2021)

DeepBlueAI achieved high performance in sub-tasks 1a and 2. This team used stacked transformer models, which used the majority vote (in the case of classification) or the average prediction (for regression) from a RoBERTa and an ALBERT model. They optimized the performance of these PLMs with a number of techniques. First, they employed task-adaptive fine-tuning (Gururangan et al., 2020) by continuing pre-training on the text of the Ha-

Team	All	Kaggle	Twitter
DeepBlueAI	0.4120	0.7607	0.2647
mmmm	0.4190	0.7757	0.2677
HumorHunter	0.4230	0.7742	0.2765
abcbpc	0.4275	0.7942	0.2712
fdabek	0.4406	0.7915	0.2979
stevenhuahua	0.4454	0.8019	0.2999
megatron	0.4456	0.8021	0.3001
MagicPai	0.4460	0.8113	0.2948
ES-JUST	0.4467	0.8065	0.2993
SarcasmDet	0.4469	0.8264	0.2861
<i>baseline (BERT)</i>	<i>0.5769</i>	<i>1.0141</i>	<i>0.4042</i>
<i>baseline (SVM)</i>	<i>0.6415</i>	<i>1.0908</i>	<i>0.4710</i>

Table 8: Results of the top performing systems for participants of task 2 (offense rating), showing RMSE for whole test set, for kaggle texts only and tweets only.

Hackathon data. They then augmented the dataset by using pseudo-labelling to generate labels for the test set, and added these to the training data. Then, after encoding the input, they used adversarial training (Miyato et al., 2016), e.g. the addition of perturbations to the embedding layer, to improve generalization. The predictions were produced after Multi Sample Dropout was applied. This approach achieved third place in task 1a and first place in task 2.

### 6.2.2 abcbpc (Pang et al., 2021)

This team deployed ERNIE 2.0 in a multi-task setup with task-specific gradients and loss for each sub-task. Using a cross-validation approach, they fine-tuned their model on each fold of data and took the average, or majority decision of their best-performing models as their predictions. Experiments demonstrated that their multi-task setup performed better than single-task learning with ERNIE 2.0, and they achieved the best score in task 1b.

### 6.2.3 Humor@IITK (Gupta et al., 2021)

This team also experimented with single-task and multi-task learning on pre-trained language models. They implemented two ensembling methods: in the single-task setup, they concatenated the embeddings produced by BERT, RoBERTa, ERNIE 2.0, DeBERTa and XLNET. In the multi-task setup, they used vote-based classification, or a weighted aggregate of outputs for the regression tasks. They also implemented an ensemble comprising a weighted average of best single-task and multi-task models, which achieved third place on task 1b. Interestingly, this team’s experiments on data augmentation, e.g. generating slightly different variations of the input sentences, disimproved performance. The team hypothesize that the impact of both humor and offense often hinges on the choice of specific words, and replacing these words with synonyms may undermine the humorous or offensive effect.

### 6.2.4 SarcasmDet (Faraj and Abdullah, 2021)

For tasks 1a, 1b and 2, this team used either BERT or RoBERTa models with different hyperparameters, and used an ensemble of these models to make predictions with hard (e.g. majority or average) voting. Interestingly, for task 1c, in which they placed third, they used a rule, that if the humor rating predicted for a text was greater or equal to 3, they labelled the text as controversial.

### 6.2.5 HumorHunter (Xie et al., 2021)

This team used DeBERTa with an embedding table which took into account the relative position of each token in the sentence. In an error analysis, they noted that texts with a question and answer were more often misclassified as humorous, possibly because this mimics the structure of a setup and punchline.

### 6.2.6 Others

PALI and stce, the top-ranking teams in task 1a, both used an ensemble of RoBERTa large, and ERNIE 2.0, but declined to submit a paper outlining further details. Similarly, the team named mmmm, which placed 2nd in both task 1b and 1c, did not furnish details of their approach.

## 6.3 Trends

### 6.3.1 Domain Adaptation

Given that the majority of the data was sourced from Twitter, several teams implemented domain adaptation strategies at different stages of their pipeline. YoungSheldon (Sharma et al., 2021) used the Ekphrasis (Baziotis et al., 2017) toolkit, which is designed for Twitter-specific preprocessing. DLJUST (Al-Omari et al., 2021) also used it in their preprocessing pipeline, and found that this achieved better results, when used in combination with some further manual spelling correction.

Domain-specific models also showed some performance improvements. UPB (Smădu et al., 2021) used BERTweet (Nguyen et al., 2020), a transformer-based language model trained on tweets for their embedding layer, and DLJUST found that this model gave slightly better performance than RoBERTa on subtask 1a, but not on the regression tasks.

Amherst685 (Gugnani et al., 2021) used intermediate fine-tuning to adapt a series of pre-trained models to the style of language used in humorous and offensive texts. They used two large humor datasets, and two offense datasets, to adapt a variety of transformer models to the task, however, they did not see performance gains from this. Similarly to DeepBlueAI, RoMa (Labadie et al., 2021) and IITH (Raha et al., 2021) used task-adaptive pre-training, and the latter team saw performance improvements of 1-5%.

### 6.3.2 Data Augmentation/Perturbation

Similarly to DeepBlueAI, MagicPai (Ma et al., 2021) experimented with pseudo-labelling in order

to increase the amount of data available. MagicPai also tried adversarial training by adding perturbations to the embedding layer, and along with Grenzlinie (Liu and Zhou, 2021) and UPB, found this to improve their transfer learning models' performance. Amherst685 tried backtranslation in order to generate more sample texts, however they found that this was not successful.

### 6.3.3 Contrasting Models and Task Setup

The majority of teams who contrasted RNNs with PLMs found that the latter was more suited to this task. ES-JUST (Bashabsheh and Alasal, 2021) found that RoBERTa performed better than RNNs and BERT. This finding replicates the ablation study by Morishita et al. (2020) in the 2020 SemEval task, which also demonstrated that RoBERTa performed better than other PLMs. However Tsia (Guan, 2021) found that RoBERTa was better suited to the regression task, and combining BERT+CNN gave better performance on the classification task. This contrasts with YoungSheldon, who achieved their best results with BERT-Base. Across all cases, we did not observe a single dominant architecture, indicating that the choice of hyperparameters and task setup played a large role in the results achieved by each team. However, teams like CS-UM6P (Essefar et al., 2021), who contrasted single and multi-task learning setups, found that the latter improved performance.

## 6.4 Other notable approaches

DUTH (Karasakalidis et al., 2021) produced a rigorous examination of different preprocessing approaches applied to data given to linear and neural models. They achieved an impressive 12th place on task 1b, with a combination of Light Gradient Boosting Machine (LGBM), XGBoost and Bayesian Ridge. They also achieved 12th place in task 1c using a combination of features such as POS-tagging, numerical features, a bigram term frequency inverse document frequency (TF-IDF) vectorizer as input to an LGBM model.

The utility of TF-IDF features was also seen in the transfer learning approaches as team hub also found that adding TF-IDF features improved the performance of their ALBERT/BERT+CNN models.

IITH found that including lexical features such as letter and punctuation counts, named entities marking, identifying personal pronouns, wh-words and question marks, as well as a lexicon of hurtful

words (Hurtlex, Bassignana et al., 2018) improved the performance of their task-adaptively pre-trained RoBERTa model for detecting humor and predicting the rating, but that only the Hurtlex features improved offense detection, and neither of these improved controversy prediction.

## 7 Analysis and Discussion

### 7.1 Correlations between Tasks

As Table 9 indicates, humor rating is moderately correlated with humor controversy across the dataset. There are no discernible trends in offense rating and humor controversy. Interestingly, there is a moderate negative correlation between humor and offense rating overall, but this is not significant for the Twitter data, and becomes a much stronger negative correlation when we look at just the Kaggle data. This may have been a factor in the finding that multi-task setups tended to achieve better results than single-task systems. It may also suggest that in naturally occurring data, such as the Twitter texts, the relationship between humor and offense may be more subtle, and therefore more difficult to detect.

Task 1	Task 2	Overall	Twitter	Kaggle
Humor	Humor	0.15	0.14	0.18
Rating	Controversy	$p = 0.0001$	$p = 0.003$	$p = 0.009$
Offense	Humor	0.07	0.11	-0.02
Rating	Controversy	$p = 0.06$	$p = 0.028$	$p = 0.82$
Humor	Offense	-0.156	-0.03	-0.42
Rating	Rating	$p = 0.0001$	$p = 0.51$	$p = 0.0011$

Table 9: Correlations between tasks, Pearson’s  $r$  and  $p$ -value

### 7.2 Differences between Kaggle Texts and Tweets

As seen in tables 5, 6 and 7, the systems’ performance for subtasks 1a, 1b and 1c seems to be consistently better for Kaggle texts than for tweets. One possible reason why systems are better at predicting humor from Kaggle texts, is that the Kaggle test set contains almost all humorous texts, while only about half of the tweets are considered humorous.

On the other hand, performance for task 2 is consistently better (lower RMSE) for tweets than for Kaggle texts, and the differences are sometimes very large. We noticed the distributions of offense ratings between Kaggle texts and tweets are very different, with tweets being more often classified

as not offensive: more than 60% of the tweets have 0.1 offense rating or less (in a scale from 0 to 5), while less than 10% of the Kaggle texts do. This difference in distribution might in part come from differences in sampling methods, because some Kaggle texts were specifically selected to have certain offensive categories, while the tweets were selected at random. In order to check if the difference in scores could come from the difference in offense rating distributions, we resampled a subset of tweets from the Kaggle set and another one from the Twitter set, trying to keep a uniform offense rating distribution, and calculated task 2 scores for those subsets. The difference between scores for these new subsets was much lower for all teams, and even some of the teams got better scores for the Kaggle subset, which might be an indication that the sharp differences in score were caused by the difference in distributions.

### 7.3 Error Analysis: Humans and Machines vs Irony

Several interesting issues arise when analyzing the top-ten systems’ errors. Irony continues to be a challenging problem, both at the annotation side, and the classification side. Several texts which were sourced from humorous accounts, and which had just less than a majority of annotator votes for humorous were classed as not-humorous in our dataset. In the following two examples, all of the top-10 systems classed this as humorous, and arguably, they are intended to be humorous, even though the majority of annotators technically did not class them as such.

1. What do you call a homosexual man on a wheel chair?  
A human being
2. It’s almost like I gotta keep myself busy with random things like fluffing pillows just so I don’t over eat.

The first example is an ironic subversion of a homophobic joke, using incongruity to undermine the anticipated punchline. While it is possible that the setup and punchline structure is what misled the system, similar question and answer structures were correctly classified.

The second example is arguably sarcasm, and all of the top systems classified it as humor, even though the annotators did not. However, there were several other texts which were classed as humorous

by the annotators, and which demonstrate traits of irony or sarcasm, were difficult to classify for the top teams, and produced mixed results:

1. If alcohol influences short-term memory, what does alcohol do?
2. How much should I rest between sets at the gym? I've been doing anywhere between 60 to 90 days to give my muscles a good chance to recover.

In terms of tasks 1b and 2, we analyzed the texts which proved most difficult to predict the humor and offense ratings for the top-10 systems. We calculated the mean average error (MAE) between the top 10 systems' predictions and the ground truth. We then examined the 75th percentile of MAE.

	Twitter	Kaggle
Humor	70%	30%
Offense	55.2%	44.8%

Table 10: Percentage of texts with highest MAE from the different sources

Interestingly, there was a disproportionately high number of Kaggle texts among the offensive texts whose rating was difficult to predict (44.8% while the Kaggle text make up only 20% of the data). A quick examination of these texts revealed there was a large number of ironic texts which were predicted to be highly offensive, although the ground truth did not reflect this, for example:

Why do black people eat fried chicken?  
Because it tastes good.

#### 7.4 Humor Controversy

As we were interested in the rule-based approach that team SarcasmDet took for this task, we investigated the upper-bound of success for any threshold-based heuristic which determines whether a text was controversial given the humor score alone. Figure 2 shows the hypothetical F1-score and accuracy that could be achieved by such a system. Assuming a perfect score on humor rating prediction, if teams assigned a controversial label for any text with a humor rating of over 2, they could achieve first place in this task in terms of accuracy with a score of 0.580. A threshold of 1.45 given perfect knowledge of the humor labels would result

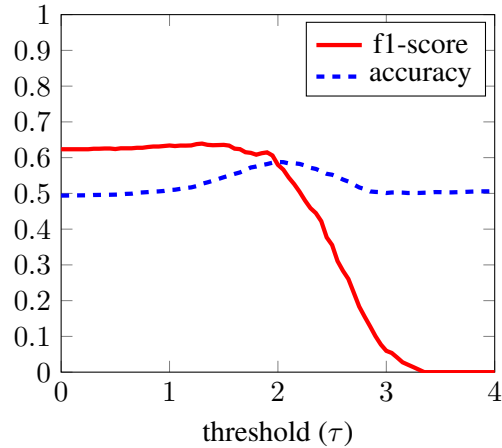


Figure 2: For varied values of a threshold,  $\tau$ , accuracy and f1-score achieved by a hypothetical model predicting the label *controversial* for all texts in the test set with ground-truth humor score  $> \tau$ . Note that participants did not have access to these ground-truth scores for the test set, making these results an upper-bound for this type of threshold-based approach.

in a leaderboard-topping F1-score of 0.635. However, the teams that took part did not obtain the perfect humor rating scores required for this simple rule to work so effectively, yet were still able to achieve similar scores on the task. This suggests that their systems were learning something, but that ultimately the task is a difficult one.

Although we aimed to increase inter-annotator agreement in this task's annotation procedure, by matching the origin of the texts and annotators, the agreement on humor ratings was low, and indeed the task which aimed to capture this controversy proved difficult.

## 8 Conclusion

We provided 10,000 texts annotated for humor and offense by a broad range of annotators. Transformer models were a dominant approach to this task, with the exception of the humor controversy task, which proved to be difficult for most teams, and in which a simple, rule-based system achieved one of the top-3 scores. As multi-task learning setups proved more effective than single-task learning demonstrates, this that there is some correlation between humor and offense detection. It was also interesting to note which model adaptations were useful and which were not. Finally, an analysis of the errors in humor analysis reveals some types of humor which may be captured inaccurately, even by the most powerful models.

## Acknowledgments

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh. The authors also wish to thank William J. Toner who acted as a last-minute Idea Bouncer.

## References

- Hani Al-Omari, Isra'a AbedulNabi, and Rehab Duwairi. 2021. DLJUST at SemEval-2021 Task 7: Hahackathon: Linking Humor and Offense Across Different Age Groups. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Miriam Amin and Manuel Burghardt. 2020. [A survey on approaches to computational humor generation](#). In *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 29–41, Online. International Committee on Computational Linguistics.
- Salvatore Attardo. 2008. A Primer for the Linguistics of Humor. *The Primer of Humor Research*, 8:101–156.
- Emran Al Bashabsheh and Sanaa Abu Alasal. 2021. ES-JUST at SemEval-2021 Task 7: Detecting and Rating Humor and Offensive Text Using Deep Learning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurltlex: A multilingual Lexicon of Words to Hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.
- Christos Baziotis, Nikos Pelekis, and Christos Doukieridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.
- Santiago Castro, Luis Chiruzzo, and Aiala Rosá. 2018. Overview of the HAHA Task: Humor Analysis Based on Human Annotation at IberEval 2018. In *IberEval@ SEPLN*, pages 187–194.
- Luis Chiruzzo, Santiago Castro, Mathias Etcheverry, Diego Garat, Juan José Prada, and Aiala Rosá. 2019. Overview of HAHA at IberLEF 2019: Humor Analysis based on Human Annotation. In *IberLEF@ SEPLN*, pages 132–144.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.
- Kabil Essefar, Abdellah El Mekki, Abdelkader El Mahdaouy, NABIL El Mamoun, and Ismail Berrada. 2021. CS-UM6P at SemEval-2021 Task 7: Deep Multi-Task Learning Model for Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Dalya Faraj and Malak Abdullah. 2021. SarcasmDet at SemEval-2021 Task 7: Detect Humor and Offensive based on Demographic Factors using RoBERTa Pre-trained Model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Zhengyi Guan. 2021. Tsia at SemEval-2021 Task 7: Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Akshay Gugnani, Brian Zylich, Gabriel Brookman, and Nicholas Samoray. 2021. Amherst685 at SemEval-2021 Task 7: Joint Modeling of Classification and Regression for Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Aishwarya Gupta, Avik Pal, Bholeshwar Khurana, Lakshay Tyagi, and Ashutosh Modi. 2021. Humor@IITK at SemEval-2021 Task 7: Large Language Models for Quantifying Humor and Offensiveness. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. [Don't stop pretraining: Adapt language models to domains and tasks](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv preprint arXiv:2006.03654*.
- Tim Highfield. 2015. Tweeted Joke Lifespans and Appropriated Punchlines: Practices around Topical Humor on Social Media. *International Journal of Communication*, 9:22.

- Jennifer Hofmann, Tracey Platt, Chloe Lau, and Jorge Torres-Marín. 2020. Gender Differences in Humor-Related Traits, Humor Appreciation, Production, Comprehension, (Neural) Responses, Use, and Correlates: A Systematic Review. *Current Psychology*, pages 1–14.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. SemEval-2020 Task 7: Assessing humor in Edited News Headlines. *arXiv preprint arXiv:2008.00304*.
- Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. *arXiv preprint arXiv:1801.06146*.
- Alexandros Karasakalidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2021. DUTH at SemEval-2021 Task 7: Is Conventional Machine Learning for Humorous and Offensive Tasks enough in 2021? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Giselinde Kuipers. 2015. The Humor Divide: Class, Age and Humor Styles. In *Good Humor, Bad Taste*, pages 71–101. De Gruyter Mouton.
- Roberto Labadie, Mariano Rodriguez, Reynier Ortega, and Paolo Rosso. 2021. Dual Transformer for Sentiment Analysis. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*.
- Renyuan Liu and Xiaobing Zhou. 2021. Grenzlinie at SemEval-2021 Task 7: HaHackathon Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Sharon Lockyer and Michael Pickering. 2005. *Beyond a Joke: The Limits of Humour*. Springer.
- Jian Ma, ShuYi Xie, Jiang Lianxin, Ryan Stark, Mo Yang, and Jianping Shen. 2021. MagicPai at SemEval-2021 Task 7: Method for Detecting and Rating Humor Based on Multi Task Adversarial Training. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- J. A. Meaney. 2020. Crossing the line: Where do demographic variables fit into humor detection? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 176–181, Online. Association for Computational Linguistics.
- Rada Mihalcea and Carlo Strapparava. 2005. Making Computers Laugh: Investigations in Automatic Humor Recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538.
- Rada Mihalcea and Carlo Strapparava. 2006. Learning to Laugh (Automatically): Computational Models for Humor Recognition. *Computational Intelligence*, 22(2):126–142.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial Training Methods for Semi-supervised Text Classification. *arXiv preprint arXiv:1605.07725*.
- Terufumi Morishita, Gaku Morio, Shota Horiguchi, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 task 8: Simple but effective modality ensemble for meme emotion recognition. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1126–1134, Barcelona (online). International Committee for Computational Linguistics.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.
- Chao Pang, Xiaoran Fan, Weiyue Su, Xuyi Chen, Shuo-huan Wang, Jiayang Liu, Xuan Ouyang, Shikun Feng, and Yu Sun. 2021. abc4pc at SemEval-2021 Task 7: ERNIE-based Multi-task Model for Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. SemEval-2017 Task 6: #Hashtagwars: Learning a Sense of Humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57.
- Tathagata Raha, Ishan Sanjeev Upadhyay, Radhika Mamidi, and Vasudeva Varma. 2021. IIITH at

- SemEval-2021 Task 7: Leveraging Transformer-based Humorous and Offensive Text Detection Architectures using Lexical and Hurltex Features along with Task Adaptive Pretraining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- J. T. Rayz. 2017. In Pursuit of Human-Friendly Interaction with a Computational System: Computational Humor. In *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMII)*, pages 000015–000020.
- Willibald Ruch. 2010. *The Sense of Humor: Explorations of a Personality Characteristic*, volume 3. Walter de Gruyter.
- Mayukh Sharma, Ilanthenral Kandasamy, and Vasantha W B. 2021. YoungSheldon at SemEval-2021 Task 7: Fine-tuning Is All You Need. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Leandro Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the Targets of Hate in Online Social Media. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 10.
- Răzvan-Alexandru Smădu, Dumitru-Clementin Cercel, and Mihai Dascalu. 2021. UPB at SemEval-2021 Task 7: Adversarial Multi-Task Learning for Detecting and Rating Humour and Offence. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Bingyan Song, Chunguang Pan, Shengguang Wang, and Zhipeng Luo. 2021. DeepBlueAI at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with Stacking Diverse Language Model-Based Methods. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Christina A Sue and Tanya Golash-Boza. 2013. ‘It Was Only a Joke’: How Racial Humour Fuels Colour-Blind Ideologies in Mexico and Peru. *Ethnic and Racial Studies*, 36(10):1582–1598.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A Continual Pre-training Framework for Language Understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8968–8975.
- Julia Taylor and S Attardo. 2017. Computational Treatments of Humor. *The Routledge Handbook of the Linguistics of Humor*. New York: Routledge, pages 456–471.
- Julia M Taylor and Lawrence J Mazlack. 2004. Computationally Recognizing Wordplay in Jokes. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26.
- Yubo Xie, Junze Li, and Pearl Pu. 2021. HumorHunter at SemEval-2021 Task 7: Humor and Offense Recognition with Disentangled Attention. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Renxian Zhang and Naishi Liu. 2014. Recognizing Humor on Twitter. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 889–898.



## A Appendices

Table 11 displays the sources for the Twitter data, e.g. 80% of the texts

Username	Count	Username	Count
humurous1liners	924	BlkMentalHealth	37
joeljeffrey	692	mikewickett	35
UberFacts	632	BlackLoveAdvice	35
Dadsaysjokes	541	JNFUSA	35
GreysAnatomyMsg	402	JokesMemesFacts	34
ConanOBrien	340	MissyDuckWife	32
boonaamohammed	337	blackbodyhealth	32
Demented_Jokes	325	RobBenedict	31
thenatewolf	284	Boyfriend_Tips	30
DailyHealthFact	284	TheJimMichaels	29
Kasandd	219	realGpad	29
songs_lyrics	203	EverBestFilms	27
Shen_the_Bird	187	NicoleB_MD	23
BadJokeCat	130	iGirlfriendTip	23
OURSELVES_BLACK	129	Grindr	23
SupereeeGO	124	MNateShyamalan	23
Mr_Truth_Hurts	112	kecia_ali	20
GayAdvicer	112	RobbyActually	19
Wizdomstweets	103	hardwick	19
TrippAdvice	102	RabbiHarvey	19
JensenAckles	97	taylorswift13	18
BunAndLeggings	93	PGATOURWives	17
MovieQuotesPage	90	tomhanks	15
annehelen	87	BlackGirlsSmile	15
YaGayAunties	83	curtisisbooger	11
mindykaling	74	evanmarckatz	11
RyanSeacrest	70	bosshogswife	11
murrman5	59	PenguinBooks	10
TheOkraProject	59	GuyStuffAdvice	10
benyahr	57	gaystarnews	10
thatonequeen	55	DrakeGatsby	9
ZaraRahim	52	offensivefcker	9
Oprah	52	outmagazine	9
michaelstrahan	43	therapy4bgirls	8
youknowwhenshe	42	ProBonoASL	4
Blackkidsswim	40	TheAdvocateMag	3
andreaavsmoak	40		

Table 11: Twitter sources of data and number of texts sourced from each account

Table 12 shows the results of the top system for each team and for each task.

Team	Task1a F1	Task1a Acc	Task1b RMSE	Task1c F1	Task1c Acc	Task2 RMSE
PALI	<b>0.9854</b>	<b>0.9820</b>	-	<b>0.6302</b>	0.4943	0.9710
stce	0.9797	0.9750	-	-	-	-
DeepBlueAI	0.9676	0.9600	0.5607	0.6257	0.4650	<b>0.4120</b>
SarcasmDet	0.9675	0.9600	0.5446	0.6270	0.4699	0.4560
mengyuan_jiayi	0.9667	0.9590	0.5621	0.5814	0.5106	-
stevenhuahua	0.9666	0.9580	0.5831	0.4991	0.5626	0.4454
zain	0.9663	0.9580	0.5748	-	-	-
EndTimes	0.9655	0.9570	0.6539	0.6261	0.4602	0.4691
MagicPai	0.9653	0.9570	0.5572	-	-	0.4460
Meizizi	0.9653	0.9570	0.6136	-	-	-
mmmm	0.9647	0.9560	0.4977	0.6279	0.4699	0.4190
fdabek	0.9647	0.9560	0.5271	0.6233	0.4537	0.4406
Isra	0.9640	0.9550	-	-	-	-
DLJUST	0.9633	0.9540	0.5555	0.4813	0.5480	0.4822
IITH	0.9616	0.9530	0.5263	0.6242	0.4537	0.4772
megatron	0.9612	0.9520	0.6307	-	-	0.4456
CS-UM6P	0.9606	0.9510	0.6360	0.6242	0.4537	0.4759
Amherst685	0.9604	0.9510	0.5339	0.4842	0.5220	0.4530
MLXG	0.9590	0.9490	2.1883	0.0000	0.5463	0.9587
abcbpc	0.9587	0.9480	<b>0.4959</b>	0.6242	0.4537	0.4275
StoneOpen	0.9583	0.9480	0.5470	0.5427	0.5561	0.4489
Humor@IITK	0.9581	0.9480	0.5210	0.6209	0.4520	0.4607
Ferryman	0.9581	0.9480	0.5651	0.6242	0.4537	0.4813
RoMa	0.9576	0.9480	0.5905	0.6197	0.4732	0.4532
HumorHunter	0.9572	0.9480	0.5510	0.6111	0.4764	0.4230
RedwoodNLP	0.9571	0.9460	0.5580	0.4883	0.5024	0.7229
UPB	0.9566	0.9470	0.6200	0.0000	0.5463	0.5318
ES-JUST	0.9564	0.9460	0.5709	0.4888	0.5545	0.4467
DeathwingS	0.9563	0.9460	0.5561	-	-	-
zeus_yao	0.9557	0.9450	-	-	-	0.4621
apostaremczak	0.9544	0.9440	0.8497	0.0000	0.4341	0.5625
LeoJ	0.9543	0.9430	2.1883	0.0000	0.5463	0.9587
CHAOYUDENG	0.9538	0.9410	-	-	-	-
gerarld	0.9532	0.9420	0.5393	0.4972	<b>0.5659</b>	0.4489
CS-UM6P	0.9506	0.9380	0.6360	0.6242	0.4537	0.4759
CSECU-DSG	0.9496	0.9380	0.6803	0.4423	0.5366	0.5395
YoungSheldon	0.9468	0.9330	0.5257	0.6210	0.4780	0.4500
DuluthNLP	0.9399	0.9260	0.6461	-	-	0.5059
pakawat.nk	0.9386	0.9240	0.5700	0.4683	0.5496	0.5368
Grenzlinie	0.9386	0.9250	0.6312	0.5455	0.5203	0.4761
bousselham	0.9368	0.9200	-	-	-	-
hub	0.9364	0.9210	0.6288	0.5591	0.5333	0.5027
ZYJ	0.9348	0.9210	0.7214	0.4603	0.4407	0.5204
xjh	0.9345	0.9180	0.6385	0.5205	0.5447	0.5151
Gulu	0.9341	0.9190	0.7405	0.5488	0.5561	0.5807
chenshi	0.9328	0.9160	0.6303	0.5547	0.5301	0.5422
UMUteam	0.9325	0.9160	0.8847	0.5722	0.4650	0.8740
Han_Jiawei	0.9286	0.9120	0.5577	0.4904	0.5268	0.5187

Zehao_Liu	0.9241	0.9060	-	-	-	-
Team KGP	0.9233	0.9030	0.5694	0.5628	0.5301	0.5800
Tsia	0.9205	0.8960	0.7010	0.4271	0.5593	0.5419
chilai1996	0.9177	0.8970	2.1883	0.0000	0.5463	0.9587
ayushnanda14	0.9081	0.8840	2.1883	0.0000	0.5463	0.9587
DUTH	0.8942	0.8720	0.5507	0.5990	0.4732	0.5819
<i>baseline</i>	<i>0.8840</i>	<i>0.8570</i>	<i>0.8609</i>	<i>0.4624</i>	<i>0.4374</i>	<i>0.6415</i>
LOLASING	0.8704	0.8490	-	-	-	0.7106
CHaines	0.8504	0.8170	0.5762	0.6242	0.4537	0.6473
AlviIshmam	0.8489	0.8160	-	-	-	-
milad.sayadamooz	0.6290	0.5270	2.5497	0.0000	0.5463	0.9587
FII Funny	0.0630	0.0780	0.5598	0.4752	0.5008	0.4788
Paima	-	-	0.5701	-	-	0.4655
abhideepmitra	-	-	1.0343	0.5366	0.4612	-
justglowing	-	-	-	-	-	0.6347

Table 12: Top system for each participant for all subtasks.

# LangResearchLab\_NC at SemEval-2021 Task 1: Linguistic Feature Based Modelling for Lexical Complexity

**Raksha Agarwal, Niladri Chatterjee**  
Indian Institute of Technology Delhi  
Hauz Khas, Delhi-110016, India  
raksha.agarwal@maths.iitd.ac.in  
niladri@maths.iitd.ac.in

## Abstract

The present work aims at assigning a complexity score between 0 and 1 to a target word or phrase in a given sentence. For each Single Word Target, a Random Forest Regressor is trained on a feature set consisting of lexical, semantic, and syntactic information about the target. For each Multiword Target, a set of individual word features is taken along with single word complexities in the feature space. The system yielded the Pearson correlation of 0.7402 and 0.8244 on the test set for the Single and Multiword Targets, respectively.

## 1 Introduction

Presence of complex words can lead to poor comprehension of a text. Identification of such complex words in a given text is a core component in the task of Automatic Simplification and Evaluation (Shardlow, 2013). The Lexical Complexity Prediction Task of SemEval 2021 (Shardlow et al., 2021) aims at development of systems for prediction of complexity scores for a target word/phrase in a given sentence. In literature, binary classification of target words in a text into complex or non-complex is referred to as Complex Word Identification (CWI) (Paetzold and Specia, 2016; Zampieri et al., 2017; Gooding and Kochmar, 2018; AbuRa'ed and Saggion, 2018; Yimam et al., 2018). Unlike previous works, a continuous complexity score is assigned to the target word in the present task which is referred to as Lexical Complexity Prediction (LCP) (Shardlow et al., 2020). For the present work, regression is performed for LCP on a set of linguistic features covering semantic, syntactic and contextual aspects of the target word as described in Section 3. Additionally, various lexicon based features are used to indicate the rarity of target words. The system achieves 0.8194 Pearson correlation for Single Word Target and 0.7482 for Multiword Target on the trial set.

## 2 Task Setup

The task is divided into two subtasks, namely Single Word Target and Multiword Target based on the length of the target. The dataset and evaluation metrics are described below.

- **Dataset:** The dataset consists of an augmented version of CompLex (Shardlow et al., 2020). It comprises sentences from three corpora, viz. World English Bible Translation, English Portion of the European Parliament proceedings, and articles from CRAFT corpus belonging to biomedical domain. It is split into three subsets Train, Trial, and Test.
- **Evaluation Metrics:** The systems are evaluated using Pearson correlation coefficient (P), Spearman rank correlation coefficient (S), Mean absolute error (MAE) and Coefficient of Determination ( $R^2$ ).

## 3 Features

In this section we present the details of the feature space used in the present work.

### 3.1 Corpus Features

A feature, named Corpus, is used to indicate to which of the 3 corpora the input sentence belongs.

### 3.2 Shallow Features

Word level shallow features used in the present work are number of letters (Nlet), syllables (Nsyl), vowels (Nvow), percentage of upper case alphabets (PerUp), simple universal part-of-speech tag (POS), and detailed Penn part-of-speech tag (Tag) of the target word extracted using SpaCy.

### 3.3 NLTK WordNet Features

Number of hypernyms (Nhyper) and number of morphemes (Nmorph) of the target word consider-

ing its POS tag in the given sentence are also used as features.

### 3.4 Exquisite Corpus (EC) Features

Exquisite Corpus<sup>1</sup> compiles texts from seven different domains namely Wikipedia, Subtitles, News, Books, Web, Twitter and Reddit. We have used the frequency (WordFreq) in EC and the Zipf frequency (ZipfFreq) of the target word as features (van Heuven et al., 2014).

### 3.5 SUBTLEX Features

Frequency (SubtFreq) of the target word extracted from SUBTLEXus<sup>2</sup> and its Contextual Diversity (ConDiversity) i.e. percent of the films in which the word appears are used as features.

### 3.6 Language Model (LM) Features

Given an input sentence  $S = w_1 w_2 \dots w_N$  and a target word  $w_t$  where  $t \in 1, 2, \dots, N$ , the following features are extracted from a trigram language model trained on the Gigaword corpus<sup>3</sup>.

- Perplexity of the input sentence (Perplexity) computed as:  

$$Perplexity(S) = \sqrt[N]{1/P(w_1 w_2 \dots w_N)}$$
- The phrase score (PhrScore) of  $w_j \dots w_t \dots w_k$  defined as  $\log_{10} P(w_j \dots w_t \dots w_k)$  where  $j = \max(1, t - 2)$  and  $k = \min(N, t + 2)$
- Average of conditional probabilities involving the target word (AvgCP)

$$Avg \left( \begin{array}{l} P(w_t | w_{t-1}, w_{t-2}), \\ P(w_{t+1} | w_t, w_{t-1}), \\ P(w_{t+2} | w_{t+1}, w_t) \end{array} \right)$$

### 3.7 Character Language Model (CharLM) Feature

The probability of the target word (Prob3c) calculated using trigram character language model is considered as a feature. The trigram<sup>4</sup> probabilities are calculated using letter counts from Google Web Trillion Word Corpus. Suppose a word  $W$  consist of  $N$  letters,  $W = w_1 \dots w_N$  then, the corresponding feature value will be computed as:  

$$Prob3c(W) = \frac{1}{N-2} \sum_{i=1}^{N-2} \log_{10} P(w_i w_{i+1} w_{i+2})$$

<sup>1</sup><https://pypi.org/project/wordfreq/>

<sup>2</sup>[https://github.com/Wonderlic-AI/wonderlic\\_nlp](https://github.com/Wonderlic-AI/wonderlic_nlp)

<sup>3</sup>lm\_giga\_64k\_nvp\_3gram.zip

<sup>4</sup>[http://norvig.com/ngrams/count\\_3l.txt](http://norvig.com/ngrams/count_3l.txt)

### 3.8 Psycholinguistic Features

The following features are extracted using MRC psycholinguistic database (Wilson, 1988): Age of acquisition (AOA), Concreteness (CONC), Imageability (IMAG) and Meaningfulness ratings (MeanC, MeanP) of the target word.

### 3.9 Kucera and Francis (KF) Features

The features derived by Kučera and Francis (1967), namely target word's written frequency of occurrence (KFFreq) and the number of categories of text in which the target word was found (KFNcats) are used.

### 3.10 Ogden Feature

A binary feature is used to indicate presence of the target word in the list of 1000 words included in Ogden's Basic English<sup>5</sup> (IsOgden).

### 3.11 Inquirer Tag Features

The General Inquirer classifies about 7500 words using 182 General Inquirer categories developed for social science content analysis (Stone et al., 1966). A binary feature is created for each category to indicate its occurrence for the target word. The POS tag of the target is matched with the 'OthTags' category to filter out incompatible categories as given in Table 1

POS of the Target	Compatible OthTags
NOUN   PRON   PROPN	NOUN   PRON
VERB   AUX   ADV	VERB   SUPV

Table 1: Inquirer Tags Filtering

## 4 Single Word Target

In the Single Word Target task, complexity scores between 0 to 1 needs to be assigned for a target word of the input sentence. Various regression models are trained using the optimal set of features using scikit-learn<sup>6</sup>. The results are presented in Table 2. For both Decision Tree and Extra Tree Regressors the maximum depth (maxdepth) is tuned between 1 to 20, and the optimal maxdepth is found to be 6 and 8, respectively. Random Forest Regressors with the default setting produced the best results for the trial dataset. Using the above, our submission to the shared task achieved 0.7402 Pearson correlation on the test set.

<sup>5</sup><http://ogden.basic-english.org/>

<sup>6</sup><https://scikit-learn.org/stable/>

Regressor	P	S	MAE	R <sup>2</sup>
Decision Tree	0.761	0.699	0.069	0.58
Extra Tree	0.757	0.650	0.071	0.57
Gradient Boosting	0.794	0.731	0.065	0.63
Random Forest	<b>0.819</b>	<b>0.748</b>	<b>0.062</b>	<b>0.69</b>
+Bagging	0.805	0.738	0.064	0.64
+Adaptive Boosting	0.798	0.734	0.065	0.63

Table 2: Results on the Trial Set

#### 4.1 Feature Importance

The Gini importance of the top 5 features are reported in Table 3. Gini importance of a feature is computed as the (normalized) total reduction of the mean squared error brought by that feature. The importance of the features is also analyzed by removing a set of features at a time and training a Random Forest Regressor for the reduced feature space. Each of the features from the optimal feature space has a positive effect on the performance of the system as indicated in Table 4. The experiments indicate that exclusion of Exquisite Corpus features led to the maximum decline in the results. Hence, this may be considered as the most important feature subset.

Feature	Gini importance
ConDiversity	0.443
Prob3c	0.072
ZipfFreq	0.068
Perplexity	0.067
AvgCP	0.060

Table 3: Gini Importance of Features

##### 4.1.1 Inquirer Tags Importance

The effect of inclusion of Inquirer Tags in the feature space has a positive effect however the magnitude is low. This may be due to the low coverage of these features as reported in Table 5. The coverage is defined as the percentage of target words having at least one Inquirer Tag.

#### 4.2 Additional Features

The following set of features when included in the feature space led to a decrease in performance for the present task on the trial set.

- Etymological Feature: The ISO code of the target word’s origin language

Features	P	S	MAE	R <sup>2</sup>
All	<b>0.819</b>	<b>0.748</b>	<b>0.062</b>	<b>0.67</b>
w/o Ogden	0.816	0.744	0.063	0.66
w/o Inquirer	0.815	0.744	0.063	0.66
w/o KF	0.815	0.746	0.063	0.66
w/o WordNet	0.814	0.747	0.063	0.66
w/o Psych	0.813	0.740	0.063	0.66
w/o LM	0.810	0.744	0.063	0.65
w/o CharLM	0.806	0.747	0.064	0.65
w/o Corpus	0.798	0.740	0.065	0.63
w/o SUBTLEX	0.795	0.725	0.066	0.63
w/o Shallow	0.786	0.728	0.067	0.61
w/o EC	0.782	0.713	0.067	0.61

Table 4: Feature Set Elimination Results for the Trial Set

Data	All	Bible	Biomed	Europarl
Train	21.14	20.23	21.48	21.77
Trial	22.09	23.78	19.26	23.08
Test	23.77	19.79	27.34	24.06

Table 5: Inquirer Tags Coverage

- Named Entity Feature: The named entity tag<sup>7</sup> of the target word.

Post task evaluation on the test set indicates their inclusion improves the performance of the system. (See Table 6)

#### 4.3 BERT Features

BERT was introduced in (Devlin et al., 2019), and its usage has resulted in state-of-the-art performance for various downstream NLP tasks, such as Question Answering, Textual Entailment and Paraphrase detection. In the present work, BERT embedding for the target word is extracted from the pre-trained BERT-base-uncased model<sup>8</sup>. Additionally, in an effort to enhance the contextualized BERT embeddings (Agarwal et al., 2020), the embedding vector is supplemented with the feature vector corresponding to linguistic features described in Section 3. Finally, a Neural Network is trained to minimize the Mean Absolute Error using Adam optimizer (Kingma and Ba, 2015). Hyper parameter tuning is performed using hyperas<sup>9</sup> and TPE algorithm. The number of intermediate dense layers are tuned between {2, 3}. The encoding dimensions are tuned between {50, 100,

<sup>7</sup>extracted using <https://spacy.io/api/entityrecognizer>

<sup>8</sup>uncased\_L-12\_H-768\_A-12.zip

<sup>9</sup><https://pypi.org/project/hyperas/>

Included in Feature Space		Trial				Test			
Etymology	NamedEntity	P	S	MAE	R <sup>2</sup>	P	S	MAE	R <sup>2</sup>
No	No	<b>0.8194</b>	<b>0.7478</b>	<b>0.0624</b>	<b>0.6681</b>	0.7402	0.7005	0.0661	0.5440
Yes	No	0.8115	0.7451	0.0633	0.6565	<b>0.7421</b>	<b>0.7013</b>	<b>0.0660</b>	<b>0.5486</b>
No	Yes	0.8113	0.7440	0.0631	0.6561	0.7404	0.6966	0.0661	0.5464
Yes	Yes	0.8175	0.7466	0.0627	0.6654	0.7418	0.6974	0.0659	0.5475

Table 6: Results for Additional Features

200, 300, 500, 700, 1000} and dropouts between {0.1, ..., 0.9}. Batch size is set to 16. The results are presented in Table 7. It can be observed that BERT embeddings do not improve the performance. Moreover, Neural Networks when applied on just linguistic features have a lower performance than Random Forest Regressors.

#### 4.4 Error Analysis

Error analysis indicates that absolute error for 87% test samples were less than 0.10. Samples belonging to Biomedical corpus had highest errors. Some predictions of the proposed model are presented in Table 8. The correlation between the actual and predicted complexity for similar targets in dissimilar contexts is high. However, it is revealed that difference in complexity of proper noun targets in distinct contexts could not be captured effectively through the present set of linguistic features.

### 5 Multiword Target

In the present task the Multiword Targets are pairs of two adjacent words. We have experimented with two approaches for predicting complexity scores for Multiword Targets, as described in Section 5.1 and Section 5.2

#### 5.1 Single Word Combination

In this approach, each word of a Multiword target is considered as individual single word targets, and the complexity scores are predicted using the Single Word Target<sup>10</sup> model. The individual word scores are combined using Average, Maximum, and Minimum. Additionally, Algebraic Sum ( $a + b - ab$ ) and Product ( $ab$ ) of the individual scores are also considered. These are taken from Fuzzy s-norm and t-norm (Klir and Yuan, 1995). The results are indicated in Table 9. For both trial and test set, maximum of the complexity score of each word of the multiword target gives the least MAE and the highest  $R^2$  value. But, the highest P

for trial set is obtained when algebraic sum of the individual complexity scores are taken and highest S is obtained when product of the individual complexity scores are taken. For the test set, algebraic sum gives highest P and S.

#### 5.2 Feature Combination

In this approach features corresponding to the individual words are concatenated, and then a regression model is trained with the increased feature space for complexity prediction. The individual target word complexity value predicted by the Single Word Target model is also considered as a feature. The results are presented in Table 10. Bagging and Adaptive Boosting are applied on Random Forest. The results indicate that inclusion of individual complexity scores enhances the performance of the system, and the best results are obtained for Bagging ensemble. Our submission to the shared task was derived using Bagging on the Random Forest Regressor. The feature set contains individual word features along with complexity scores. It achieved Pearson correlation of 0.8244 on the test set.

### 6 Conclusion

Identification of difficult words is an important task for Automatic Text Simplification. LCP aims at assigning scores to words of a given sentence to indicate its complexity. In this work we utilize word level features to capture its lexical, semantic and syntactic information. LM based features are used for indicating the semantics of the target word in a given context. Frequency and occurrence based features are used to indicate the overall rarity of the target words. For Single Word Target, Random Forest Regressors trained on the linguistic feature set achieved the highest results. Error analysis revealed that the model can be further improved to capture the context of the target word.

For Multiword Target, two approaches were explored. In the first approach complexity scores of individual target words predicted by the Sin-

<sup>10</sup>Random Forest Regressor w/o additional features

Feature Set	# Dense Layers	Dimension	Dropouts	P	S	MAE	R <sup>2</sup>
Linguistic	2	50,200	0.1, 0.3	0.752	0.698	0.070	0.563
BERT	3	300,300,1000	0.3,0.1,0.1	0.732	0.678	0.071	0.532
BERT + Linguistic	2	300,200	0.1,0.1	0.714	0.660	0.072	0.502

Table 7: Results on Trial Set for Neural Network

Input Sentence	Target Word	Actual Complexity	Predicted Complexity
Saul arose, and they went out both of them, he and Samuel, abroad.	Saul	0.3676	0.3398
Saul said to his servants, "Provide me now a man who can play well, and bring him to me."	Saul	0.3529	0.3383
Samuel said to Saul, "Why have you disturbed me, to bring me up?"	Saul	0.2778	0.3303
These results, as well as this study, suggest that a considerable amount of maternal cholesterol can be transferred to the murine fetus.	amount	0.2031	0.2048
This wild-type staining pattern may simply reflect the fact that decreasing the amount of mutant protein by half makes it undetectable by immunocytochemistry.	amount	0.2375	0.2207

Table 8: System Predictions

Combination Strategy	Trial				Test			
	P	S	MAE	R <sup>2</sup>	P	S	MAE	R <sup>2</sup>
Average	0.7329	0.7239	0.1220	0.0437	0.8098	0.8101	0.1314	0.0110
Maximum	0.6872	0.6733	<b>0.1021</b>	<b>0.2861</b>	0.7907	0.7916	<b>0.1041</b>	<b>0.3433</b>
Minimum	0.6964	0.6970	0.1534	-0.4056	0.7036	0.7064	0.1648	-0.5466
AlgebraicSum	<b>0.7391</b>	0.7217	0.1270	0.0598	<b>0.8193</b>	<b>0.8104</b>	0.1049	0.3349
Product	0.7047	<b>0.7298</b>	0.3153	-3.8253	0.7704	0.8063	0.3257	-3.9447

Table 9: Results for Multiword Target for Single Word Combination

Individual Complexity Predictions as a feature	Regressor	P	S	MAE	R <sup>2</sup>
No	Random Forest	0.7327	0.7253	0.0885	0.5110
	+Bagging	0.7299	0.7294	0.0877	0.5118
	+Adaptive Boosting	0.7386	0.7369	0.0880	0.5167
Yes	Random Forest	0.7234	0.7256	0.0872	0.5134
	+Bagging	<b>0.7482</b>	<b>0.7510</b>	<b>0.0830</b>	<b>0.5517</b>
	+Adaptive Boosting	0.7455	0.7427	0.0853	0.5408

Table 10: : Results for Multiword Target on the Trial Set using Feature Combination

gle Word model were combined using different strategies, while in the second, the feature space was expanded to accommodate features and complexity scores corresponding to individual target words. The latter yielded the best results. Our sys-

tem achieved 36<sup>th</sup> and 17<sup>th</sup> rank with respect to the two subtasks. The difference in the correlation value between the top performer is less than 0.05 for Single Word Target and 0.04 for Multiword Target.



## Acknowledgments

Raksha Agarwal acknowledges Council of Scientific and Industrial Research (CSIR), India for supporting the research under Grant no: SPM-06/086(0267)/2018-EMR-I

## References

- Ahmed AbuRa'ed and Horacio Saggion. 2018. [LaS-TUS/TALN at complex word identification \(CWI\) 2018 shared task](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–165, New Orleans, Louisiana. Association for Computational Linguistics.
- Raksha Agarwal, Ishaan Verma, and Niladri Chatterjee. 2020. [LangResearchLab\\_NC at FinCausal 2020, task 1: A knowledge induced neural net for causality detection](#). In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 33–39, Barcelona, Spain (Online). COLING.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications@NAACL-HLT 2018, New Orleans, LA, USA, June 5, 2018*, pages 184–194. Association for Computational Linguistics.
- Walter J. B. van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. [Subtlex-uk: A new and improved word frequency database for british english](#). *Quarterly Journal of Experimental Psychology*, 67(6):1176–1190. PMID: 24417251.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- George Klir and Bo Yuan. 1995. *Fuzzy sets and fuzzy logic*, volume 4. Prentice hall New Jersey.
- Henry Kučera and Winthrop Nelson Francis. 1967. *Computational analysis of present-day American English*. University Press of New England.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Philip J Stone, Dexter C Dunphy, and Marshall S Smith. 1966. *The general inquirer: A computer approach to content analysis*. MIT press.
- Michael Wilson. 1988. Mrc psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior research methods, instruments, & computers*, 20(1):6–10.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. [Complex word identification: Challenges in data annotation and system performance](#). In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63, Taipei, Taiwan. Asian Federation of Natural Language Processing.

# SINAI at SemEval-2021 Task 1: Complex word identification using Word-level features

**Jenny Ortiz-Zambrano**  
Universidad de Guayaquil  
Guayaquil, Ecuador  
jenny.ortizz@ug.edu.ec

**Arturo Montejo-Ráez**  
CEATIC - Universidad de Jaén  
Jaén, España  
amontejo@ujaen.es

## Abstract

This article describes a system to predict the complexity of words for the Lexical Complexity Prediction (LCP) shared task hosted at SemEval 2021 (Task 1) with a new annotated English dataset with a Likert scale. Located in the Lexical Semantics track, the task consisted of predicting the complexity value of the words in context. A machine learning approach was carried out based on the frequency of the words and several characteristics added at word level. Over these features, a supervised random forest regression algorithm was trained. Several runs were performed with different values to observe the performance of the algorithm. For the evaluation, our best results reported a M.A.E of 0.07347, M.S.E. of 0.00938, and R.M.S.E. of 0.096871. Our experiments showed that, with a greater number of characteristics, the precision of the classification increases.

## 1 Introduction

The identification of complex words (CWI) is the task of detecting in the content of documents the words that are difficult or complex to understand by the people of a certain group (Rico-Sulayes, 2020). The CWI and the substitution of words identified as complex may significantly improve readability and understandability of a given text (Zotova et al., 2020).

CWI has become an area of great interest in recent years for the computational linguistics community in making proposals that allow researchers to develop computational semantic analysis systems, as demonstrated by the shared tasks of CWI in SemEval 2016 (Paetzold and Specia, 2016), y NAACL-HTL 2018 (Yimam et al., 2018), and the CWI task of the ALexS 2020 competition, hosted at IberLEF 2020 (Ortiz-Zambrano and Montejo-Ráez, 2020).

This article introduces a system that has participated in the Lexical Complexity Prediction (LCP) shared task hosted at SemEval 2021 (Task 1) (Shardlow et al., 2021a). The task releases a new annotated English dataset with a Likert scale. Located in the Lexical Semantics track, the task consisted of predicting the complexity value of the words in context.

We have explored different features for representing words and multi-words and their context. Some preprocessing steps have been evaluated along with the effect of feature selection.

## 2 Related Work

(DuBay, 2004) defines readability as allowing one text to be easier to read than another. For many people, the understanding of a text can be affected by the presence of lexically and semantically complex words and phrases, for example for children (Petersen and Ostendorf, 2009), non-native speakers (Petersen and Ostendorf, 2009), and people with various cognitive or reading disabilities (Saggion et al., 2015).

Predicting which words a given target population has difficulty to understand is a critical step for many NLP applications, such as in text simplification, which has traditionally focused its attention on second language learners, native speakers with low levels of literacy, and people with language disabilities reading (Saggion et al., 2015). This task is also known as complex word identification (CWI). The prediction of the lexical complexity carried out with precision can allow to adapt texts according to the needs of the readers (Shardlow et al., 2020). Actually, in an early study in the 1920s, a very simple way to predict the level of difficulty of a text was discovered by educators, who used vocabulary difficulty and sentence length as main indicators (DuBay, 2004).

corpus	bible	europarl	biomedic	total
single	2574	2576	2512	7662
multi	505 1	498	514	1517

Table 1: Total number of sentences in each training corpus.

### 3 Dataset

The training data set provided to the participants consisted of an augmented version of CompLex (Shardlow et al., 2021b). It uses data from three different sources: the Bible, Europarl, and biomedic texts (see Table 1). It is a set of multidomain English data made up of sentences, the targeted token, and its respective level of complexity as described in (Shardlow et al., 2020).

### 4 The system

This section describes the details of the system applied to the task, as our approach to complex word identification. A machine learning approach was followed based on the frequency of the words and further characteristics added at word level. Over these features, a supervised random forest regression algorithm was trained. In this section, first, the features considered in the supervised learning approach are introduced. Then, the method to determine whether a candidate word is complex or not is detailed.

#### 4.1 Features

We computed a total of 15 features, taking into consideration the linguistic measures of the work carried out by (Mc Laughlin, 1969) and the experiments of the shared tasks of the CWI BEA 2018 respectively by (Paetzold and Specia, 2016; Gooding and Kochmar, 2018). These are the features obtained on the target word (token).

- *Absolute frequency (abs-frequency)*: the absolute frequency. This frequency is computed based on the unannotated corpora compiled by José Cañete<sup>1</sup> from different sources. It contains about 3 billion words.
- *Relative frequency (rel-frequency)*: the relative frequency of the target word.
- *Word length (length)*: the number of characters of the token.

<sup>1</sup>Available at <https://github.com/josecannete/spanish-corpora>

- *Number of syllables (number-syllables)*: the number of syllables.
- *Target word position (token-position)*: the position of the target word in the sentence.
- *Number of words in the sentence (n-words-sentences)*: number of words in sentence.
- *Part Of Speech (POS)*: the Part Of Speech category.
- *Relative frequency of the previous the token (freq-rel-word-before)*: the relative frequency of the word before the token.
- *Relative frequency of the word after the token (freq-rel-word-after)*: the relative frequency of the word after the token.
- *Length of previous word (len-word-before)*: the number of characters in the word before the token.
- *Length of the after word (len-word-after)*: the number of characters in the word after the token.
- *Measure of Textual Lexical Diversity (MTLD-diversity)*: the lexical diversity of the target word in the sentence using the metric proposed by (McCarthy and Jarvis, 2010)<sup>2</sup>.

Additionally, the following WordNet (Fellbaum, 2010) features were also considered for each target word:

- *Number of synonyms (number-synonyms)*.
- *Number of hyponyms (number-hyponyms)*.
- *Number of hyperonyms (number-hyperonyms)*.

In the case of multiple words, the following characteristics were applied: absolute frequency, relative frequency, token length, number of syllables, total number of words in the sentence, MTDL diversity.

<sup>2</sup>Computed using this Python library: <https://pypi.org/project/lexical-diversity/>

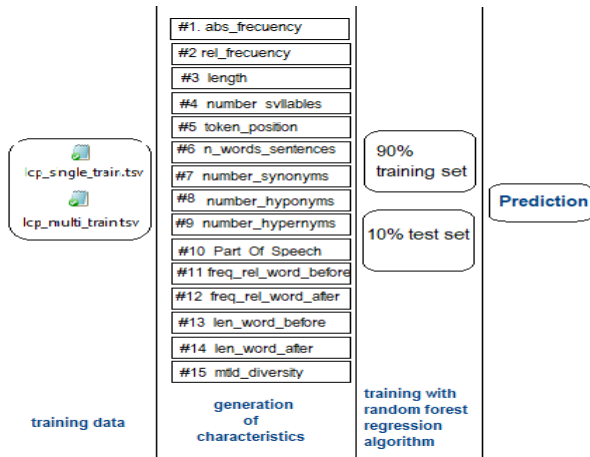


Figure 1: Training process applying the Random Forest Regression algorithm. A different model is trained for each training subset of data.

## 4.2 Method

The numeric input variables were scaled to a standard range, as many machine learning algorithms have been found to perform better when the data set is normalized. A polynomial transformation on the features characteristics was then applied with a degree value of 2, so new features were created.

A forest of trees was built with the training set  $(X, y)$ , where we assigned to the independent variable  $(X)$  an array that contains all the word-level characteristics that were obtained from the token, the same ones that were described in the section 4.1; and the value of the dependent variable  $(y)$  corresponds to the level of complexity’s word.

To build the Random Forest Regression Model, we split the dataset into the training set and test set, that is, 10% of the data set was used as test set, and the remaining 90% was used as the training set. Figure 1 shows the training process applying the random forest regression algorithm.

## 5 Experimental Results

### 5.1 Results on Trial and Simulated Data

To calculate the prediction value of the word complexity on the data of the evaluation corpus, the  $(X, y)$ , where we assigned to the independent variable  $(X)$  which we called  $X_{Test}$ , was built, was an array that contained all the word-level characteristics that were obtained from the token. Finally, we train the algorithm with the evaluation data and predict the results of the test set with the model trained on the testing set values using the regressor predict

#Trees	K	MAE	MSE	RMSE
150	7	0.07347	0.00938	0.09687
130	7	0.07354	0.00940	0.09700
150	8	0.07356	0.00942	0.09710

Table 2: Results obtained with Random Forest with selecting K-best features on single words subset.

#	Team Name	MAE	MSE	$R^2$
1	JUST_Blue	0.0609	0.0062	0.6172
2	DeepBlueAI	0.0610	0.0061	0.6210
3	Alejandro M.	0.0619	0.0064	0.6062
50	SINAI	0.0875	0.0131	0.1930

Table 3: Final results of the Lexical Complexity Prediction task on the single words dataset

function.

Several runs were made with different values to observe the performance of the algorithm and fine-tune the hyperparameters of the model.

Our best configuration was with 150 nodes and 7 features, selected by their F ANOVA between label / feature. The selected characteristics were: abs\_frequency, rel\_frequency, length, number\_syllables, token\_position, number\_synonyms, Part\_of\_speech. Finally, the prediction value of the words for the test data set was obtained, obtaining the best result: MAE of 0.07347, MSE of 0.00938, and RMSE of 0.096871 (see Table 2).

### 5.2 Results on test Data

In this section we present the results obtained from our system, and we carry out a discussion regarding the results presented by the organizers of the workshop.

The final results were sent to the SemEval 2021 organizers after the execution of our system. The final published results are those shown in Table 3, where the winners of the first three positions are presented. The results that we obtained in the contest for the case of the evaluation corpus of simple words were, MAE of 0.0875, MSE of 0.0131 and R-squared of 0.1930. Taking into account the number of competitors (quite large) and the result obtained by the first place winner (MAE of 0.0609), we see that there is a small difference, which allows us to be confident with our simple approach.

## 6 Conclusion

In this article, the results of our participation in Task 1: Lexical Complexity Prediction in the Lexical semantics track hosted at the SemEval 2021 international workshop have been presented. Both the training corpus and the evaluation corpus were provided by the sponsoring organization of this competition. We applied machine learning and built the model using the random forest regression algorithm, relying on well-known word based and contextual features.

As future work, we plan to perform error analysis on the predictions, to identify the weaknesses of the proposed approach based on a characterization of the instances where the system performs poorly. Also, a better analysis of multi-word scenario is foreseen.

## Acknowledgments

We appreciate , Luis Alexander Suárez Colimba, graduates of the Computer Systems Engineering degree from the University of Guayaquil, for their valuable contribution to the development of our work.

This study is partially funded by the Spanish Government under the LIVING-LANG project (RTI2018-094653-B-C21).

## References

- William H DuBay. 2004. The Principles of Readability. *Online Submission*.
- Christiane Fellbaum. 2010. WordNet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.
- Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- G Harry Mc Laughlin. 1969. SMOG grading-a new readability formula. *Journal of reading*, 12(8):639–646.
- Philip M McCarthy and Scott Jarvis. 2010. MTL, vocd-D, and HD-D: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior research methods*, 42(2):381–392.
- Jenny A Ortiz-Zambrano and Arturo Montejo-Ráezb. 2020. Overview of ALexS 2020: First Workshop on Lexical Analysis at SEPLN.
- Gustavo Paetzold and Lucia Specia. 2016. [SemEval 2016 Task 11: Complex Word Identification](#). pages 560–569.
- Sarah Petersen and Mari Ostendorf. 2009. [A machine learning approach to reading level assessment](#). *Computer Speech Language*, 23:89–106.
- A Rico-Sulayes. 2020. General Lexicon-Based Complex Word Identification Extended with Stem Ngrams and Morphological Engines. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)*, CEUR-WS, Malaga, Spain.
- Horacio Saggion, Sanja Štajner, Stefan Bott, Simon Mille, Luz Rello, and Biljana Drndarevic. 2015. [Making It Simplex: Implementation and Evaluation of a Text Simplification System for Spanish](#). *ACM Trans. Access. Comput.*, 6(4).
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A New Corpus for Lexical Complexity Prediction from Likert Scale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting Lexical Complexity in English Texts. *arXiv preprint arXiv:2102.08773*.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. *arXiv preprint arXiv:1804.09132*.
- Elena Zotova, Montse Cuadros, Naiara Perez, and Aitor García-Pablos. 2020. Vicomtech at ALexS 2020: Unsupervised Complex Word Identification Based on Domain Frequency. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020)*, CEUR-WS, Malaga, Spain.

# TUDA-CCL at SemEval-2021 Task 1: Using Gradient-boosted Regression Tree Ensembles Trained on a Heterogeneous Feature Set for Predicting Lexical Complexity

Sebastian Gombert and Sabine Bartsch

Corpus and Computational Linguistics, English Philology

Institute of Linguistics and Literary Studies

Technische Universität Darmstadt, Germany

sebastian.gombert@outlook.de, sabine.bartsch@tu-darmstadt.de

## Abstract

In this paper, we present our systems submitted to *SemEval-2021 Task 1* on *lexical complexity prediction* (Shardlow et al., 2021a). The aim of this shared task was to create systems able to predict the *lexical complexity* of word tokens and bigram multiword expressions within a given sentence context, a continuous value indicating the difficulty in understanding a respective utterance. Our approach relies on gradient boosted regression tree ensembles fitted using a heterogeneous feature set combining linguistic features, static and contextualized word embeddings, psycholinguistic norm lexica, *WordNet*, word- and character bigram frequencies and inclusion in word lists to create a model able to assign a word or multiword expression a context-dependent complexity score. We can show that especially contextualised string embeddings (Akbik et al., 2018) can help with predicting lexical complexity.

## 1 Introduction

In this paper, we present our contribution to *SemEval-2021 Shared Task 1* (Shardlow et al., 2021a), a shared task focused on the topic of lexical complexity prediction. The term *lexical complexity prediction* describes the task of assigning a word or multiword expression a continuous or discrete score signifying its likeliness of being understood well within a given context, especially by a non-native speaker. Solving this task could benefit second-language learners and non-native speakers in various ways. One could imagine using such scores to extract vocabulary lists appropriate for a learner level from corpora and literature (Alfter and Volodina, 2018), to judge if a given piece of literature fits a learner’s skill or to assist authors of textbooks in finding a level of textual difficulty appropriate for a target audience.

Predicting these scores can be formulated as a regression problem. Our approach to solve this

problem relies on *gradient-boosted regression tree ensembles* which we fit on a heterogeneous feature set including different word embedding models, linguistic features, *WordNet* features, psycholinguistic lexica, corpus-based word frequencies and word lists. We assumed that lexical complexity could be correlated with a wide range of features, neural ones as much as distributional or psycholinguistic ones, which is why we chose to use an ensemble-based method in the form of gradient boosting (Mason et al., 1999) for our system as it usually performs best for tasks where such a feature set is needed compared to solely neural models which need dense, homogeneous input data to perform well.

Out of all participants, our systems were ranked 15/54 in the single word- and 19/37 in the multiword category during the official shared task evaluations according to *Pearson’s correlation coefficient*. Our key discovery is that while features from nearly all categories provided by us were used by our systems, *contextual string embeddings* (Akbik et al., 2018) were the by far most important category of features to determine lexical complexity for both systems. The code and our full results can be found at <https://github.com/SGombert/tudacclsemeval>.

## 2 Background

### 2.1 Task Setup

For the shared task, *CompLex* corpus (Shardlow et al., 2020, 2021b) was used as data set. This English corpus consists of sentences extracted from the *World English Bible* of the multilingual corpus consisting of bible translations published by Christodoulopoulos and Steedman (2015), the English version of *Europarl* (Koehn, 2005), a corpus containing various texts concerned with European policy, and *CRAFT* (Bada et al., 2012), a corpus

consisting of biomedical articles.

*CompLex* is divided into two sub-corpora, one dealing with the complexity of single words and the other one with the complexity of bigram multiword expressions. Accordingly, the shared task was divided into two sub-tasks, one dedicated to each sub-corpus. Within both *CompLex* sub-corpora, the sentences are organised into quadruples consisting of a given sentence, a reference to its original corpus, a selected word, respectively a multiword expression from this sentence, and a continuous complexity score denoting the difficulty of this selected word or bigram which is to be predicted by systems submitted to the shared task. For the task, both subcorpora were partitioned into training, test and trial sets.

The scores given for simple words, respectively multiword expressions, were derived from letting annotators subjectively judge the difficulty of understanding words respectively word bigrams on a Likert scale ranging from 1 to 5 with 1 indicating a very simple and 5 a very complex word. The assigned scores were then projected onto values between 0 and 1 and averaged between all annotators to calculate the final scores.

## 2.2 Related Work

The first approaches to the systematic prediction of lexical complexity were made during *SemEval-2016 Task 11* (Paetzold and Specia, 2016). Here, the problem of determining the complexity of a word was formulated as a classification task designed to determine whether a word could be considered as being complex or not. The data set used for this task was created by presenting 20 non-native speakers with sentences and letting them judge whether the words contained within these sentences were rated as complex or not. From these judgements, two different data sets were derived. In the first one, a word was considered complex if at least one of the annotators had judged it as such, and in the second one, each word was given 20 different labels, one per annotator. The most important findings for this shared task were that ensemble methods performed best in predicting lexical complexity with word frequency being the best indicator.

In 2018, a second shared task was conducted on the same topic as described in Yimam et al. (2018). This shared task focused on predicting lexical complexity for English, German, Spanish and a multi-

lingual data set with a French test set. The data for this was acquired by presenting annotators on *Amazon Mechanical Turk* with paragraphs of text and letting them mark words which according to their perception could hinder the same paragraph from being understood by a less proficient reader. The findings of this shared task confirmed the finding of the previous one that using ensemble methods yield best results for complex word identification with a system submitted by Gooding and Kochmar (2018) relying on decision tree ensembles.

## 3 System Overview

Our systems rely on *gradient-boosted regression tree ensembles* (Mason et al., 1999) for predicting lexical complexity scores. We trained one model to predict single word lexical complexity scores and another one to predict bigram multiword expression complexity scores. Our models are based on the implementation of gradient boosting provided by *CatBoost*<sup>1</sup> (Dorogush et al., 2018; Prokhorenkova et al., 2018). We set the growing policy to *loss-guide*, the *L2 leaf regularisation* to 15, the *learning rate* to 0.01, *tree depth* to 6 and the *maximum number of leaves* to 15. Additionally, we set the *number of maximum iterations* to 5000 and then used the trial set to perform *early stopping* during training in order to determine the exact number of required iterations.

The motivation behind using this algorithm was its general ability to perform well on heterogeneous and sparse feature sets which allowed us to mix regular linguistic features, *WordNet* features, word embeddings, psycho-linguistic norm lexica, corpus-based word frequencies and selected word lists as all of these were features we assumed to possibly correlate with lexical complexity. Moreover, the reportings of Paetzold and Specia (2016) and Yimam et al. (2018) that ensemble-based learners perform best for *complex word identification* contributed to this decision, as well. While the problem presented in their paper is formulated as a binary classification task using different data sets, we wanted to test if their findings would still translate to a regression task on *CompLex*.

### 3.1 Feature Engineering

The following paragraphs describe the features we used to create the feature vectors used to represent words. In case of our system dealing with bigram

<sup>1</sup><https://catboost.ai/>

multiword expressions, we calculated such a vector for each of both words and then concatenated them to acquire the final input vectors. Thus, the exact number of input features was 7424 for our system dealing with single words and 14848 for our system dealing with multiword expressions.

**Syntactic features:** This category of features includes *XPOS*-, *UPOS*-, *dependency*- and *named entity tags* as well as *universal features*<sup>2</sup> inferred using the English *Stanza*<sup>3</sup> (Qi et al., 2020) model fit to the version of the *English Web Treebank* following the *Universal Dependencies* formalism (Silveira et al., 2014). In addition to the tags assigned to the word(s) whose score was to be predicted, we included the *XPOS*- and *UPOS* tags of the two neighbouring words to the left and to the right as well as the dependency tags of the siblings, direct children and the parent of the word(s) within the dependency structure of a given sentence. All of these features are encoded as *one*-, respectively *n-hot vectors* using the *LabelBinarizer* and *MultiLabelBinarizer* classes provided by *Scikit-learn* (Pedregosa et al., 2011).

**WordNet features:** Here, we included the numbers of *hypernyms*, *root hypernyms*, *hyponyms*, *member holonyms*, *part meronyms* and *member meronyms* of the respective word(s) as well as the number of given examples and the *length of the shortest hypernym path* from *WordNet* (Miller, 1995). In cases where multiple *synsets* were given for a word, we calculated the respective means and in cases where a given word was not included in the resource, we set all respective feature values to 0. We accessed *WordNet* using *NLTK* (Bird et al., 2009). The main intuition behind using this resource was that the *length of the shortest hypernym path* and the count for the different lexico-semantic relations could be a good indicator for lexical complexity.

**Word embeddings:** We used multiple static and contextual word embedding models for our feature set. This includes the *transformer-based* (Devlin et al., 2019) *BiomedNLP-PubMedBERT-base-uncased-abstract* (Gu et al., 2020), *distilgpt2*<sup>4</sup> (Radford et al., 2018) and *distilbert-base-uncased* (Sanh et al., 2019), the *contextual string embed-*

*ding* models *mix-forward* and *mix-backward*<sup>5</sup> (Akbik et al., 2018), and the static *GloVe*<sup>6</sup> (Pennington et al., 2014) and English *fastText*<sup>7</sup> (Bojanowski et al., 2017) embeddings.

This collection of embeddings was derived from previous experiments on the *CompLex* corpus where we tried to fine-tune a purely neural model using the approach of stacking different embedding models in combination with an attached prediction head central to *flairNLP*<sup>8</sup> (Akbik et al., 2019). More precisely, in the setup we chose, the outputs of all language models were fed to a feed-forward layer responsible for calculating the final complexity scores. This network was then trained for 5 epochs with a *learning rate* of 0.000001, *mean squared error* as loss function and *Adam* (Kingma and Ba, 2015) as optimizer on the training set part of *CompLex*. During this training, fine-tuning was active for all transformer-based language models so that their weights were adjusted during the process and *scalar mixing* (Liu et al., 2019) was used for the transformer-based language models as it was not foreseeable which layers of the transformer models would influence results the most.

This model achieved a *Pearson's correlation coefficient* score of 0.7103 when evaluated on the trial set. While we deemed this an okay result, we decided to stick with gradient boosting for our final systems as early experiments with this algorithm yielded results superior to the purely neural approach when evaluated on the same set. As we switched to using gradient boosting for our final systems, we decided to use the fine-tuned variants of the transformer embedding models as using them led to small improvements when testing our models on the shared task trial sets compared to using the non-fine-tuned variants.

**Psycholinguistic norm lexica:** Our feature set includes two psycholinguistic norm lexica. The first one is described in Warriner et al. (2013) and scores words with empirical ratings for *pleasantness*, *arousal* and *dominance* using the *SAM score* (Bradley and Lang, 1994). These ratings were acquired from annotators on the *Amazon Mechanical Turk* platform. The second lexicon is described in

<sup>2</sup><https://universaldependencies.org/u/feat/all.html>

<sup>3</sup><https://stanfordnlp.github.io/stanza/>

<sup>4</sup><https://huggingface.co/distilgpt2>

<sup>5</sup>[https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/FLAIR\\_EMBEDDINGS.md](https://github.com/flairNLP/flair/blob/master/resources/docs/embeddings/FLAIR_EMBEDDINGS.md)

<sup>6</sup><https://nlp.stanford.edu/projects/glove/>

<sup>7</sup><https://fasttext.cc/>

<sup>8</sup><https://github.com/flairNLP/flair>



Malandrakis and Narayanan (2015) and includes ratings for *arousal, dominance, valence, pleasantness, concreteness, imagability, age of acquisition, familiarity, pronouncability, context availability* and *gender ladenness*. The ratings within this lexicon were derived algorithmically from smaller lexicons using linear combinations and semantic similarity scores to approximate the ratings for words not included in the source lexica. In both cases, the inclusion of these features was mainly motivated by our general intuition that the perceived complexity of words could be linked to different psycholinguistic variables.

**Word frequencies:** We utilised three resources containing corpus-based word respectively character bigram frequencies. The first of these data sets was the frequency list extracted from the *SUBTLEXus* corpus (Brysbaert and New, 2009) consisting of various movie subtitles from which we used the *log-normalised* term frequency and the *log-normalised* document frequency as features. Besides *SUBTLEXus*, we utilised the character bigram frequencies from Norvig (2013) which were extracted from the *Google Books Corpus*. Here, to represent a word, we calculated the mean of all frequencies of the bigrams constituting the same and used this as feature. In the case of both sets, our intuition was that lower frequency would likely function as a proxy for complexity. The third set we used was *EFLLex* (Dürlich and François, 2018) which lists the frequencies of words within several pieces of English literature appropriate for different *CEFR*<sup>9</sup> levels. We included this set as we deemed that *CEFR* as a framework for rating language competence could also function as an according proxy.

**Word Lists:** We used two different word lists as features. The first one is *Ogden’s Basic English Vocabulary*<sup>10</sup>, a list of simple words used for writing *simple English* as described in Ogden (1932). Here, our idea was that this could help to identify simple words within *CompLex*. The other one was the *Academic Word List* as described in Coxhead (2011), a structured lexicon of terms used primarily in academic discourse which we believed to contain more complex words. In both cases, we encoded the inclusion of a word within a respective word list binarily.

<sup>9</sup><https://tracktest.eu/english-levels-cefr/>

<sup>10</sup><http://ogden.basic-english.org/>

Metric	System	Rank	Best Res.
<b>Pearson</b>	0.7618	15/54	0.7886
<b>Spearman</b>	0.7164	26/54	0.7425
<b>MAE</b>	0.0643	20/54	0.0609
<b>MSE</b>	0.0067	9/54	0.0061
<b>R2</b>	0.5846	10/54	0.6210

Table 1: Results achieved by our system dealing with single word complexity. **Best Results** refer to the best score achieved within each category by a competing system.

Metric	System	Rank	Best Res.
<b>Pearson</b>	0.8190	19/37	0.8612
<b>Spearman</b>	0.8091	19/37	0.8548
<b>MAE</b>	0.0711	14/37	0.0616
<b>MSE</b>	0.0080	12/37	0.0063
<b>R2</b>	0.6677	13/37	0.7389

Table 2: Results achieved by our system dealing with multiword expression complexity. **Best Results** refer to the best score achieved within each category by a competing system.

## 4 Results

Throughout the shared task, the systems were evaluated with regard to *Pearson’s correlation coefficient*, *Spearman’s rank correlation coefficient*, *mean average error*, *mean squared error* and *R2* with *Pearson’s correlation coefficient* determining the main ranking. According to this, our systems achieved the 15th and 19th rank respectively. Table 1 shows the results achieved by our system dealing with single words and Table 2 the results achieved by our system dealing with multiword expressions. The results show that our systems, while only achieving upper mid-table results on average, come close to the best systems performance-wise which speaks for our approach. Further hyperparameter tuning and the addition of more features could likely close this gap. The full results for all submitted systems are presented in Shardlow et al. (2021a).

### 4.1 Most Important Features

To determine which features were used by our models to predict lexical complexity, we rely on the functionality provided by *CatBoost* which scores each feature for its influence on a given final prediction. This is achieved by changing a respective feature values and observing the resulting change

Rank	Feature	Importance
1	flair-mix-b.	25.10
2	flair-mix-b.	11.79
3	flair-mix-b.	7.03
4	flair-mix-f.	4.09
5	flair-mix-f.	2.98
6	flair-mix-b.	1.33
7	flair-mix-f.	1.20
8	distilbert-b.-u.	1.19
9	BiomedNLP	1.12
10	GloVe	1.03

Table 3: The 10 most important features observed for our system dealing with single word complexity and their categories. Each entry refers to a single dimension of the feature vector.

in the model prediction (see <sup>11</sup> for further information on the exact method). The outputs of this method are normalised so that the sum of the importance values of all features equals 100. *Feature importance* was calculated using the evaluation set of *CompLex*.

Inspecting the results of these calculations, we noticed that our systems did not use the character bigram frequencies derived from the *Google Books Corpus*, nor the frequencies from *EFLLex* or the word list inclusion features. While features from all other categories were utilised, the most dominant features by far are contained in the word embedding category. Within this category, the most dominant features for both models came from the *flair-mix-backward* and *flair-mix-forward* models (see Tables 3 and 4). A few single dimension from the embeddings provided by *flair-mix-backward* seem to play the major role here.

In the case of our model dealing with multiword expressions, the ten most important features all stem from the *flair-mix-backward* embedding of the second word. This could be explained by the fact that most multiword expressions within the *CompLex* corpus follow the structure of a semantic head in combination with a modifier as most of them are either multi token compounds or single token nouns modified by adjectives. It is intuitive from a linguistic point of view that in such cases, the semantic head, which comes as second element, should play the dominant semantic role resulting in it being more influential in the overall results.

<sup>11</sup><https://catboost.ai/docs/concepts/fstr.html>

Rank	Feature	Importance
1	flair-mix-b. (2nd w.)	9.28
2	flair-mix-b. (2nd w.)	7.24
3	flair-mix-b. (2nd w.)	6.09
4	flair-mix-b. (2nd w.)	3.80
5	flair-mix-b. (2nd w.)	3.60
6	flair-mix-b. (2nd w.)	3.17
7	flair-mix-b. (2nd w.)	2.44
8	flair-mix-b. (2nd w.)	1.88
9	flair-mix-b. (2nd w.)	1.34
10	flair-mix-b. (2nd w.)	1.08

Table 4: The 10 most important features observed for our system dealing with multiword expression complexity and their categories. Each entry refers to a single dimension of the feature vector.

While the exact reason for the strong influence of the *contextualised string embeddings* is hard to determine due to the fact that embeddings lack the property of being easily interpretable, we assume that the dominant role they play for the results could be determined by them being calculated on the character level (Akbik et al., 2018) instead of the level of fixed words or subword units such as morphemes. As a consequence, such models use fewer input dimensions and each of the dimensions present is in turn involved in the encoding of more different words. This links each input dimension also to a larger variety of latently encoded distributional knowledge which could then contain certain regularities strongly correlated with lexical complexity. However, without further research, this currently remains pure speculation.

## 4.2 Predictions vs. Ground Truth

In order to compare the predicted values of our models to the ground truth data, we scatterplotted the relationship between ground truth labels and the scores predicted by our systems (see Figures 1 and 2) using the *CompLex* evaluation set. It can be observed that both systems, especially the one dealing with single word complexity, show the tendency to assign slightly higher scores than given in the ground truth for simple words and slightly lower scores for complex words. The system dealing with multiword expressions does not assign any value below 0.2 at all and the one dealing with single word complexity rarely does so. This indicates that our feature set does not contain features which could help our models to identify very simple words.

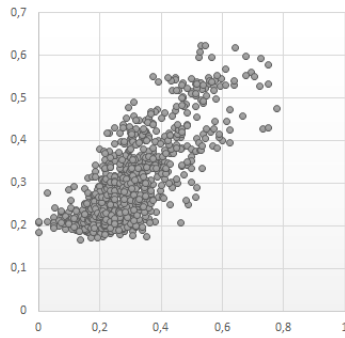


Figure 1: Scatterplot visualising the relationship between the ground truth and the predictions of our model for single word complexity. **X**: ground truth **Y**: prediction

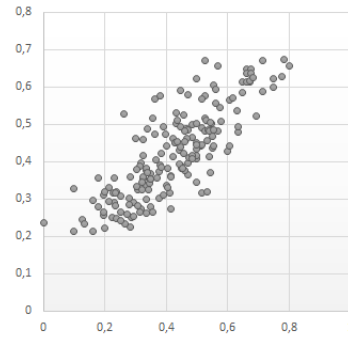


Figure 2: Scatterplot visualising the relationship between the ground truth and the predictions of our model for multiword expression complexity. **X**: ground truth **Y**: prediction

## 5 Conclusion

We presented both our systems submitted to *SemEval-2021 Task 1* combining a heterogeneous feature set with gradient boosting as regression algorithm. Our systems were ranked 15/54 and 19/37 during shared task evaluations according to *Pearson's correlation coefficient*. However, the results achieved by our systems were still close to the best results, especially in the case of the system dealing with single word complexity. The type of feature playing the most important role for our models are *contextual string embeddings* as they influenced the outcome the most. We attribute this to a relationship between lexical complexity and the distribution of characters throughout words and sentences, but this needs further clarification which could be the objective of future work. Moreover, our systems rarely assign scores below 0.2. It must be explored further if there are features which could improve our systems in this respect. In summary, we can report that ensemble methods turned out to be fruitful when applied to *CompLex*.

## References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. [FLAIR: An easy-to-use framework for state-of-the-art NLP](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. [Contextual string embeddings for sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- David Alfter and Elena Volodina. 2018. [Towards single word lexical complexity prediction](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 79–88, New Orleans, Louisiana. Association for Computational Linguistics.
- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William Baumgartner Jr, Kevin Cohen, Karin Verspoor, Judith Blake, and Lawrence Hunter. 2012. [Concept annotation in the CRAFT corpus](#). *BMC bioinformatics*, 13:161.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Margaret M. Bradley and Peter J. Lang. 1994. [Measuring emotion: The self-assessment manikin and the semantic differential](#). *Journal of Behavior Therapy and Experimental Psychiatry*, 25(1):49–59.
- Marc Brysbaert and Boris New. 2009. [Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English](#). *Behavior Research Methods*, 41(4):977–990.
- Christos Christodoulopoulos and Mark Steedman. 2015. [A massively parallel corpus: the Bible in 100 languages](#). *Lang. Resour. Evaluation*, 49(2):375–395.
- Averil Coxhead. 2011. [The academic word list 10 years on: Research and teaching implications](#). *TESOL Quarterly*, 45(2):355–362.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. [CatBoost: gradient boosting with categorical features support](#). *CoRR*, abs/1810.11363.
- Luise Dürlich and Thomas François. 2018. [EFLLex: A graded lexical resource for learners of English as a foreign language](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. [Domain-specific language model pretraining for biomedical natural language processing](#).
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Philipp Koehn. 2005. [Europarl: A Parallel Corpus for Statistical Machine Translation](#). In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.
- Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. [Linguistic knowledge and transferability of contextual representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.
- Nikolaos Malandrakis and Shrikanth S. Narayanan. 2015. [Therapy language analysis using automatically generated psycholinguistic norms](#). In *Proceedings of Interspeech*.
- Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. 1999. [Boosting algorithms as gradient descent](#). In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS’99, page 512–518, Cambridge, MA, USA. MIT Press.
- George A. Miller. 1995. [WordNet: A lexical database for English](#). *Commun. ACM*, 38(11):39–41.
- Peter Norvig. 2013. [English letter frequency counts: Mayzner revisited or ETAOIN SRHLDCU](#).
- C.K. Ogden. 1932. *Basic English: A General Introduction with Rules and Grammar*. Psyche miniatures. K. Paul, Trench, Trubner & Company, Limited.
- Gustavo Paetzold and Lucia Specia. 2016. [SemEval 2016 task 11: Complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. [Scikit-learn: Machine learning in Python](#). *Journal of Machine Learning Research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. [CatBoost: unbiased boosting with categorical features](#). In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS’18*, page 6639–6649, Red Hook, NY, USA. Curran Associates Inc.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A python natural language processing toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. [Language models are unsupervised multitask learners](#).
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter](#). *ArXiv*, abs/1910.01108.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. [Predicting lexical complexity in English texts](#). *arXiv preprint arXiv:2102.08773*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. [A gold standard dependency corpus for English](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. [Norms of valence, arousal, and dominance for 13,915 english lemmas](#). *Behavior research methods*, 45(4):1191–1207.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

# JCT at SemEval-2021 Task 1: Context-aware Representation for Lexical Complexity Prediction

Chaya Liebeskind, Otniel Elkayam and Shmuel Liebeskind

Department of Computer Science, Jerusalem College of Technology

21 Havaad Haleumi St., P.O.B. 16031

9116001 Jerusalem, Israel

(liebchaya, otniel.elkayam, israellieb)@gmail.com

## Abstract

In this paper, we present our contribution in SemEval-2021 Task 1: Lexical Complexity Prediction, where we integrate linguistic, statistical, and semantic properties of the target word and its context as features within a Machine Learning (ML) framework for predicting lexical complexity. In particular, we use BERT contextualized word embeddings to represent the semantic meaning of the target word and its context. We participated in the sub-task of predicting the complexity score of single words.

## 1 Introduction

Over the last decade, automated methods for detecting complex words have been developed. At the beginning, most of these methods assumed that lexical complexity is binary, words are either "difficult" or "not difficult". Thus, the first Complex Word Identification (CWI) shared task referred to binary identification of complex words (Zampieri et al., 2017). The main limitation of this assumption is that a word close to the decision boundary is considered to be as complex as one farther apart. Therefore, three years ago, the CWI included an additional probabilistic classification task where the participants were asked to give a probability of the given target word in particular context being complex (Štajner et al., 2018).

Recently, CompLex, a new English corpus for lexical complexity prediction was introduced (Shardlow et al., 2020). The corpus is annotated using a 5-point Likert scale (1-5) (corresponding to very easy, easy, neutral, difficult, and very difficult), and covers 3 genres: Bible translation, European Parliament proceedings, and biomedical articles. SemEval-2021 (Task 1) shared task on Lexical Complexity Prediction (LCP) (Shardlow et al., 2021a,b) provided the participants with CompLex and defined two sub-tasks: predicting the com-

plexity score of single words, and predicting the complexity score of multi-word expressions.

We present our system for the first sub-task of predicting the complexity score of single words. Our system incorporates linguistic, statistical, and semantic properties of the target word and its context as features within a Machine Learning (ML) framework for predicting lexical complexity.

This paper is organized as follows: First, in Section 2, we describe features from previous works that we have adopted. Then, in Section 3, we describe our feature sets, the feature selection process, and the results on the trial data. Finally, Our system results on the test data are detailed in Section 4, followed by conclusions in Section 5.

## 2 Related work

In this section, we shortly describe linguistic, statistical, and semantic features which were encoded as features in previous complexity prediction tasks and were integrated in our system.

Linguistics features, such as Part-Of-Speech (POS) tag, dependency parsing relations, and syllable counts, as well as statistical features, such as word length and word frequency, have been widely used for predicting lexical complexity (Mukherjee et al., 2016; Ronzano et al., 2016; Alfter and Pilán, 2018; Gooding and Kochmar, 2018; Hartmann and Dos Santos, 2018; Kajiwara and Komachi, 2018; Wani et al., 2018). Some of these works found WordNet (Miller, 1998) as a valuable source of lexical features. The main extracted feature is the number of synsets, but also information on hypernyms, hyponyms, holonym, and meronym is useful (Gooding and Kochmar, 2018; Hartmann and Dos Santos, 2018; Wani et al., 2018).

Semantic features were commonly encoded using word embedding representation of the meaning of words (Kuru, 2016; AbuRa'ed and Sag-

gion, 2018). These word embeddings were generated using Word2Vec context-independent models (Mikolov et al., 2013). Word2Vec models combine different senses of the word into one single vector. However, recently, there is a growing interest in contextualized word representations, such as BERT (Devlin et al., 2018). BERT model generates context-dependent embeddings that allow a word to have several vector representations depending on the context in which it is used. In contrast to previous works that only use context-independent embeddings, our system uses the BERT-based context-dependent embeddings.

### 3 System Description

We adopt a supervised Machine Learning (ML) approach for lexical complexity prediction. The first step in a classifier training is to determine which text characteristics are relevant and how those features are coded.

#### 3.1 Feature Sets

We next detail how the semantic properties of the sentence, as well as the linguistic and statistical properties found useful in prior work, are encoded as features. Then, in Section 3.2, we describe our feature analysis procedure and the supervised ML model. The features in our model are divided into 3 sets: linguistic, statistical and semantic.

##### 3.1.1 Linguistic features

Our dataset contains three corpora: Bible, Europarl, and Biomedical, to add variation. Since each corpus has its own unique linguistic features, we first encode the text source by three binary features.

Most of our linguistic features are based on information extracted from a POS tagger. Our linguistic properties include two families of properties: morphological and syntactical.

First, we encode the target word POS. The POS is extracted by the Spacy’s statistical POS tagger<sup>1</sup>. Each possible POS tag is represented as a binary feature. We use the following 12 tags from the Universal POS tags<sup>2</sup>: ADJ, ADP, ADV, CONJ, DET, NOUN, NUM, PRT, PRON, VERB and X (other). As an additional feature, the number of syllables in the target word is encoded<sup>3</sup>. Then,

<sup>1</sup><https://spacy.io/>

<sup>2</sup><https://universaldependencies.org/u/pos/index.html>

<sup>3</sup><https://eayd.in/?p=232>

we calculate the number of punctuation marks and stopwords in the sentence (two features).

Next, we represent syntactic forms by POS patterns. The POS pattern refers to seven words, the target word and three words before and after it. Each of the words is encoded by 12 binary features, resulting with 84 features.

We also measure the polysemy degree of the target word using the number of senses in WordNet. We obtain two lexical features: number of synsets for the target word and number of synsets for the target word given its POS.

##### 3.1.2 Statistical features

We define some statistical features based on frequency. First, we calculate target word length and sentence length. Then, we extract the target word frequency using Google N-gram<sup>4</sup> word frequencies. We encode the logarithm of this frequency as a feature to speed the ML algorithm’s convergence (three features).

##### 3.1.3 Semantic features

We represent the meaning of the surrounding context of the target word by vectors in the same semantic space. We use the BERT semantic space. BERT is a bidirectional transformer pre-trained on a large corpus containing the Toronto Book Corpus and Wikipedia using a combination of masked language modeling objective and next sentence prediction. BERT contextualizing vectors are used to represent the semantic meaning of the sentence by averaging the BERT vectors of seven words, the target word and three words before and after it. Thus, our semantic representation add 768 features (the size of BERT output layer).

To extract additional features, we use two machine learning algorithm: K-Means and k-Nearest Neighbors (KNN) algorithm. K-Means is an unsupervised learning algorithm used for clustering. It takes the unlabeled dataset and tries to group them into  $k$  number of clusters. We encode the K-Mean results by four binary features, a feature per cluster ( $k=4$ ). The results of the KNN algorithm are encoded similarly. However, KNN is a supervised learning algorithm used for classification. It takes the labeled dataset and uses it to learn how to label other sentences. KNN classifies an unseen sentence using its  $k$  nearest neighbors voting. We use four complexity classes: 0-0.25, 0.26-0.5, 0.51-0.75, 0.76-1.

<sup>4</sup><https://books.google.com/ngrams>

### 3.2 Feature Selection

For each of the above feature sets, we tried to filter out non-relevant features using several approaches.

First, we discharged features that decrease the system performance on the training set, namely, the POS pattern features, the WordNet features, and the K-Means and KNN features. We were left with 794 features. These features were selected using the Linear Regression algorithm, which was also selected as a baseline algorithm by the task organizers. To further improve the performance of our systems, we used additional ML algorithms, such as SVM and XGBoost (see more details in Section 3.3).

Next, since correlated features do not carry unique information and may interfere the learning, we tried to discharge highly correlated features. We implemented this approach using the following iterative process. The input is the desired final number of features. First, we define an initial correlation threshold (0.9). Then, we calculate the features' pairwise correlation and features with correlation above the threshold are removed. Next, if we still have more features than desired, we will lower the correlation threshold (by 10%) and repeat the process. This approach improved the performance of the SVM and Linear Regression models (selecting 97 features), but did not increase the performance of the XGBOOST method.

We note that we also tried to filter out feature using the principal component analysis (PCA) feature selection method (Song et al., 2010). PCA aims to pick a subset of features that retains as much information present in the full data as possible. PCA was performed both on the full feature list and on specific features, such as BERT features, but it was not successful.

Some of the classification models had low performance using such amount of features (794 features). Therefore, we further filleted features by calculating their correlation with the complexity score and discarding features with low correlation (less than 0.072). We resulted with the following list of 101 features:

- Biomedical corpus indicator
- Europal corpus indicator
- NOUN POS tag
- PRON POS tag

- number of syllables in the target word
- target word length
- target word frequency (Google N-gram)
- 94 features from BERT vector

It is interesting to note that even though, there are 12 POS tags, only 2 are informative for the complexity prediction task. Considering the source text indicators, the third Bible indicator is not useful. Out of the BERT 768 features, only 94 remained (12.2% of the vector).

The BERT representation of the sentence is generated by pre-trained language representation model. These models can be trained on different datasets of various domains. Since one of our corpora is from the Biomedical domain, we examined the system performance using the domain specific BioBERT (Lee et al., 2020). Figure 1 shows a comparison between the error rate of our system using the classic BERT and BioBERT (BERT on the left and BioBERT on the right). The columns show the error rate for different text sources. The red line is the average error rate. Columns from left to right: Bible, Biomedical, and Europarl. Surprisingly, the error rate of the BioBERT on the Biomedical domain is higher than that of the classic BERT. However, the average error for both is the same ( $\sim 0.69$ ).

### 3.3 Application of five Machine Learning methods

We combined the features in a supervised classification framework using five ML methods: Linear Regression, Supported Vector Machine (SVM), XGBoost (XGB), KNN, and Stacking (Stack). We trained the ML methods on the train set and evaluated their performances on the trial set.

We ran these ML methods by the scikit-learn open-source machine-learning package in python<sup>5</sup> (Pedregosa et al., 2011) using the default parameters. Table 1 shows the performances of the different ML methods on the feature set of 101 features, as described above. The MAE is omitted from the table because it is similar for all the ML algorithms (0.01). The performance differences between the algorithms were not so substantial. Therefore, we next report the performances of all these methods on the test set.

<sup>5</sup><https://scikit-learn.org/stable/>



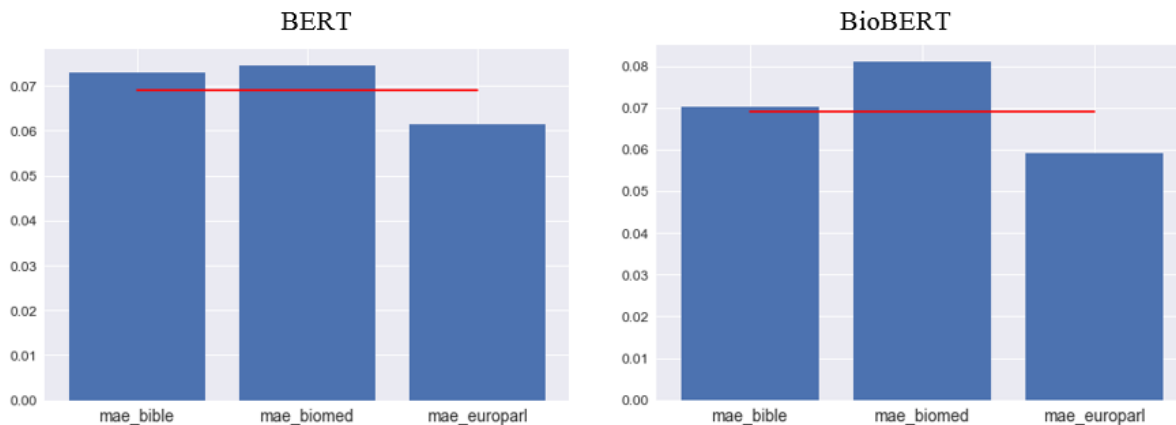


Figure 1: A comparison between the error rate of our system using the classic BERT and BioBERT

Alg.	Pearson	Spearman	MSE	R2
LR	0.672	0.656	0.077	0.45
SVM	0.693	0.67	0.075	0.476
XGB	0.671	0.655	0.076	0.449
KNN	0.682	0.64	0.077	0.464
Stack	0.689	0.66	0.077	0.436

Table 1: The performances of the different ML algorithms on the trial set

## 4 Results

To increase the size of our train set for the test phase of the task, we used both the train and trial sets to train the final model. Table 2 presents our results on the test set. The predictions of the XGBoost were submitted to the shared task competition. The results of the different algorithms are close to each other and consistent with the results on the trial set. The results of the KNN method are a bit lower. Even though, stacking allows to use the strength of each individual classifier by using their output as input of a final classifier, it did not obtain better result. This may imply that the different classifiers exploit the same information and do not reveal supplementary information.

Alg.	Pear.	Spea.	MAE	MSE	R2
LR	0.629	0.622	0.079	0.01	0.384
SVM	0.669	0.645	0.074	0.009	0.439
XGB	0.666	0.646	0.074	0.009	0.44
KNN	0.618	0.598	0.074	0.01	0.358
Stack	0.658	0.633	0.079	0.009	0.433

Table 2: The performances of the different ML algorithms on the test set

To analyze our results, we converted the complexity scores to labels following [Shardlow et al. \(2020\)](#) descriptors. In Figure 2, we present the classification confusion matrix of the XGBoost algorithm. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. Most of the classification errors (18.54%) were due to incorrect classification of very easy words as easy. There were also errors in the opposite direction (4.36%). Most of the rest of the classifications were between neutral and easy in both directions (7.42% + 6.43% = 13.85%). We note that the 5<sup>th</sup> class, very difficult, does not appear in the confusion matrix since there are not any very difficult words in the test set and the system did not classified any of the words as very difficult.

True	Predicted			
	very easy	easy	neutral	difficult
very easy	35 3.82%	170 18.54%	8 0.87%	0 0.00%
easy	40 4.36%	441 48.09%	68 7.42%	0 0.00%
neutral	1 0.11%	59 6.43%	72 7.85%	2 0.22%
difficult	0 0.00%	6 0.65%	14 1.53%	1 0.11%

Figure 2: A confusion matrix for the XGBoost complexity predictions

## 5 Conclusions and Future Work

We have implemented a system that incorporates linguistic, statistical, and semantic features to predict lexical complexity of target word in context. BERT semantic space was used to represent the word and its context. We investigated several feature selection approaches and used various supervised algorithms.

Even though our system was not highly ranked, we believe that some of the presented ideas can be useful for future research on lexical complexity prediction. In particular, we think that BERT is a powerful model that should be explored. Perhaps, fine-tuning BERT for the complexity prediction task would increase the system performance.

## References

- Ahmed AbuRa'ed and Horacio Saggion. 2018. LaS-TUS/TALN at the Complex Word Identification 2018 Shared Task. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications; 2018 Jun 5; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 159–65. ACL (Association for Computational Linguistics).*
- David Alfter and Ildikó Pilán. 2018. SB@ GU at the Complex Word Identification 2018 Shared Task. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 315–321.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Sian Gooding and Ekaterina Kochmar. 2018. CAMB at the Complex Word Identification 2018 Shared Task: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- Nathan Hartmann and Leandro Borges Dos Santos. 2018. Nilc at the Complex Word Identification 2018 Shared Task: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340.
- Tomoyuki Kajiwara and Mamoru Komachi. 2018. Complex word identification based on frequency in a learner corpus. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 195–199.
- Onur Kuru. 2016. Ai-ku at Semeval-2016 task 11: Word embeddings and substring features for complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1042–1046.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.
- George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.
- Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Ju\_nlp at Semeval-2016 task 11: Identifying complex words in a sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 986–990.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830.
- Francesco Ronzano, Luis Espinosa Anke, Horacio Saggion, et al. 2016. Taln at Semeval-2016 task 11: Modelling complex words by contextual, lexical and semantic features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1011–1016.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A new corpus for lexical complexity prediction from likert scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.
- Fengxi Song, Zhongwei Guo, and Dayong Mei. 2010. Feature selection using principal component analysis. In *2010 international conference on system science, engineering design and manufacturing informatization*, volume 1, pages 27–30. IEEE.
- Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A report on the Complex Word Identification Shared Task 2018.

*In Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications.*

Nikhil Wani, Sandeep Mathias, Jayashree Aanand Gajjam, and Pushpak Bhattacharyya. 2018. The whole is greater than the sum of its parts: Towards the effectiveness of voting ensemble classifiers for complex word identification. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 200–205.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*.

# IAPUCP at SemEval-2021 Task 1: Stacking Fine-Tuned Transformers is Almost All You Need for Lexical Complexity Prediction

**Kervy Rivas Rojas**

Research Group on Artificial Intelligence  
Pontificia Universidad Católica del Perú  
k.rivas@puccp.edu.pe

**Fernando Alva-Manchego**

Department of Computer Science  
University of Sheffield  
f.alva@sheffield.ac.uk

## Abstract

This paper describes our submission to SemEval-2021 Task 1: predicting the complexity score for single words. Our model leverages standard morphosyntactic and frequency-based features that proved helpful for Complex Word Identification (a related task), and combines them with predictions made by Transformer-based pre-trained models that were fine-tuned on the Shared Task data. Our submission system stacks all previous models with a LightGBM at the top. One novelty of our approach is the use of multi-task learning for fine-tuning a pre-trained model for both Lexical Complexity Prediction and Word Sense Disambiguation. Our analysis shows that all independent models achieve a good performance in the task, but that stacking them obtains a Pearson correlation of 0.7704, merely 0.018 points behind the winning submission.

## 1 Introduction

Complex Word Identification (CWI) consists of determining which words or multi-word expressions (MWE) in a text could be difficult to understand by certain readers. This is one of the first steps in the typical Lexical Simplification pipeline (Shardlow, 2014). CWI has traditionally been treated as either a binary (Paetzold and Specia, 2016) or regression (Štajner et al., 2018) task. For the latter, the complexity of a word/MWE was computed as a percentage of binary complexity ratings. Recently, Shardlow et al. (2020) proposed to move away from the binary definition of CWI, and instead collected complexity ratings using Likert scales. This allows re-defining the task as Lexical Complexity Prediction (LCP). Leveraging this new collected data, the First LCP Shared Task was organised in SemEval-2021 (Shardlow et al., 2021).

Our team participated in Sub-task 1: predicting the complexity score of single words. Basically, given a sentence and a target word in it, the

goal is to predict the complexity score of the target. One particular challenge is that the same target can have different complexity scores depending on the sentence it appears in. Therefore, our proposed approach takes the context of the target into consideration in two ways. First, we use contextualised word representations from pre-trained Transformer-based models, such as RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019). In particular, we use the LCP data to fine-tune two RoBERTa models and one XLNet model that receive as input the target and a context window of 1, and a RoBERTa model whose inputs are the target and a context window of 2. Second, we hypothesise that different contexts could evoke different senses of the target word. As such, we exploit data for Word Sense Disambiguation (WSD) through multi-task learning. In particular, we fine-tune a BERT (Devlin et al., 2019) model with two tasks: LCP and WSD, using the Unified Evaluation Framework (Raganato et al., 2017) for the latter. The predictions from all these models are combined with several morphosyntactic and corpus-based features, and used to train a Gradient Boosting Decision Tree with LightGBM (Ke et al., 2017).

On the test set of the Shared Task, our model achieved a Pearson correlation of 0.7704 and ranked 10th, only 0.018 points behind the winner. An ablation study shows that all independent models contributed to the stacked model’s performance, with the predictions from the BERT model fine-tuned in a multi-task fashion having the greatest impact in predicting lexical complexity. The code to reproduce our results is available in: [https://github.com/kdrivas/lexical\\_complexity](https://github.com/kdrivas/lexical_complexity).

## 2 Background

The LCP Shared Task on SemEval-2021 asks participants to develop models that predict the com-

Sentence with Target	Complexity
<i>His left hand is under my <b>head</b>.</i>	0.125
<i>Do therefore according to your wisdom, and don't let his gray <b>head</b> go down to Sheol in peace.</i>	0.383

Table 1: Annotated sentences in the dataset of the LCP Shared Task. The target word is boldfaced.

plexity of a target word/MWE in a sentence in English (Shardlow et al., 2021). This Shared Task builds on previous editions that focused on Complex Word Identification (Paetzold and Specia, 2016; Štajner et al., 2018), with a key difference: complexity ratings are continuous scores instead of binary. Furthermore, the same target word/MWE can appear in more than one sentence but with different complexity scores. Table 1 presents an example from the data.

The data for the Shared Task is an extension of CompLex (Shardlow et al., 2020), a dataset with complexity ratings for target words/MWE in sentences in English in three domains: Bible, Europarl and Biomed. The dataset is split into two subtasks: LCP for single words and LCP for MWEs.

### 3 System Description

This section details our stacking approach to the LCP Shared Task Sub-task 1. An overview of our system can be seen in Figure 1.

#### 3.1 Features

After joining all the data from both subtasks (single word and MWE), we extracted some features presented in (Yimam et al., 2018; Finnimore et al., 2019) and other custom ones, such as (1) the complexity of the target words in the lexicon proposed in (Maddela and Xu, 2018), (2) the predictions from four fine-tuned Transformer based models, and (3) the number of senses and dependencies of the target word/MWE.

##### 3.1.1 Morphosyntactic and Lexical Features

First, we computed the number of characters and the number of words surrounding the target word/MWE. In addition, we obtained the part-of-speech of the first token and the syntactic dependencies of the whole target using the spaCy library.<sup>1</sup> We also counted the number of possible part-of-speech tags for the token using the Brown dictio-

<sup>1</sup><https://spacy.io/>

nary in NLTK.<sup>2</sup> Then, we counted the number of propositions, verbs, nouns, adverbs and got the ratio between the number of nouns and verbs using the whole sentence. Finally, we calculated the total number of syllables and morphemes.

##### 3.1.2 N-gram Features

We formed n-grams considering one and two tokens surrounding the target word/MWE. Then, we computed their frequency in the Children’s Book Test (Hill et al., 2015) and Simple Wikipedia (Kauchak, 2013). In addition, using the previous corpora, the Lang-8 corpus (Mizumoto et al., 2011) and the Tatoeba corpus,<sup>3</sup> we computed the frequency of the target tokens.

##### 3.1.3 Word Complexity Lexicon

The lexicon created in (Maddela and Xu, 2018) contains complexity scores for more than 15,000 words. After lower-casing the words in the lexicon and the datasets from the Shared Task, we assigned the complexity from the lexicon to the words in the LCP data. If the word does not appear in the lexicon we assigned a null value.

##### 3.1.4 Transformer-based Model Predictions

The last set of features is composed of the predictions of four pre-trained language models fine-tuned on the training data of both subtasks. The first three were a RoBERTa (Liu et al., 2019) and an XLNet (Yang et al., 2019) models that received as input the target word/MWE and a context window of 1, and a RoBERTa model with the target and a context window of 2. The last model was a BERT fine-tuned in a multi-task fashion with two tasks: LCP and Word Sense Disambiguation (WSD). For the former task, we only used the data generated with a window size of 1 and, for the latter, the Unified Evaluation Framework (Raganato et al., 2017).

**Multi-Task Model.** Given a sentence  $S$  of the dataset of the Shared Task and a complex word  $w$  in position  $a$  whose part of speech is  $p$ , we obtain a subsequence of size 1,  $sub = \langle w_{a-1}, w_a, w_{a+1} \rangle$ ; then:

$$CLS = BERT(sub) \quad (1)$$

where  $CLS$  is the  $CLS$  token of BERT, which

<sup>2</sup><https://www.nltk.org/>

<sup>3</sup>Available in <https://tatoeba.org/> under a CC-BY 2.0 FR licence.

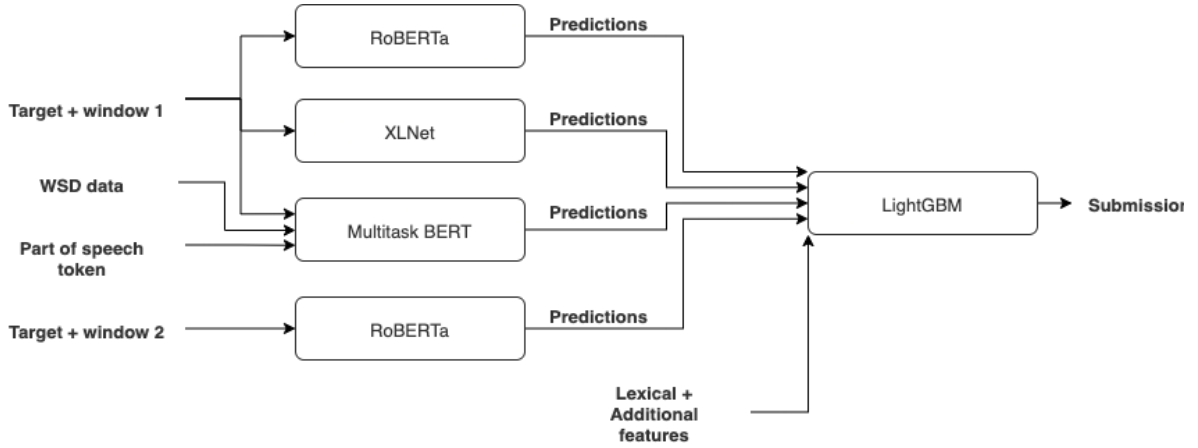


Figure 1: We used a LightGBM on the top of our architecture. It received the additional features and, the predictions from a XLNet and a BERT models using a window size of 1 and two RoBERTa models using a window size of 1 and 2.

represents the sentence. This representation is concatenated with the embedding token of  $p$ :

$$c = \text{concat}(CLS, \text{embed}(p)) \quad (2)$$

The concatenated vector is then used as input to a dropout layer and a linear layer:

$$\text{out}_1 = \text{Linear}(\text{Dropout}(c)) \quad (3)$$

Using  $\text{out}_1$ , we computed loss  $L_1$  using mean squared error. After getting the first task loss, we computed the loss for the second one. Given an ambiguous sentence  $S$  and a sequence output of senses id  $A$ , we used the BertForTokenClassification implementation in HuggingFace<sup>4</sup> to obtain the output  $\text{out}_2$ , and then used cross entropy to compute loss  $L_2$ . Finally, we multiply a weight per each task loss to get the final overall loss:

$$L = W_1 * L_1 + W_2 * L_2 \quad (4)$$

Finally, we perform other experiments

### 3.2 Architecture

Our model architecture is shown in Figure 1. First, we got the predictions from the four language models. Then, we concatenated those predictions with the additional features, and stacked a LightGBM model that received them as input features.

<sup>4</sup>[https://huggingface.co/transformers/model\\_doc/bert.html#bertfortokenclassification](https://huggingface.co/transformers/model_doc/bert.html#bertfortokenclassification)

## 4 Experimental Setup

As previously described, we used four different models: RoBERTa, XLNet, BERT and LightGBM. In addition, for training/fine-tuning each model we chose the Mean Absolute Error (MAE) as our validation metric.

### 4.1 RoBERTa and XLNet

We fine-tuned the models for 4 epochs with a batch size of 24. In addition, we used a learning rate of  $2e-5$  and Adam optimizer. We used the models for sequence classification provided by HuggingFace.<sup>5</sup>

### 4.2 Multitask BERT

We fine-tuned a BERT model using two tasks: LCP and WSD. We trained the WSD task using the Unified Evaluation Framework (Raganato et al., 2017), but filtered sentences with a size greater 22 tokens. For fine-tuning, we used a learning rate of  $2e-5$  and Adam optimizer. We fine-tuned the models for 5 epochs with a batch size of 32. We calculated the loss accumulating the gradients from both tasks. Also, we experimented with assigning different weights to each task, and found that the best configuration was 0.8 for LCP and 0.2 for WSD.

### 4.3 LightGBM

At the top of our architecture, we used a LightGBM model. Using Hyperopt, a bayesian optimization framework, we set up a max depth of 5, num-leaves

<sup>5</sup>[https://huggingface.co/transformers/model\\_doc/roberta.html#robertaforsequenceclassification](https://huggingface.co/transformers/model_doc/roberta.html#robertaforsequenceclassification)

of 8, min-sum-hessian-in-leaf of 0.9, a bagging-fraction of 0.9, a bagging-freq of 100, a learning-rate of 0.08, and a min-data-per-group of 100. We trained using 500 iterations with an early stopping of 90. Also, we declared the type of corpus and the part of speech as categorical features.

## 5 Results

The test set contains more than 1,000 sentences with 573 different target words. Table 2 shows the official evaluation metrics for each domain-corpora in the LCP dataset. Overall, we achieved a Pearson correlation of 0.7704, and finished in 10th place in the Shared Task Sub-task 1, only 0.018 points behind the winning submission.

Corpus	Pearson	Spearman	MAE	MSE
Bible	0.7536	0.7300	0.064	0.0074
Europarl	0.7492	0.7028	0.052	0.0045
Biomed	0.7898	0.7608	0.070	0.0083
Overall	0.7704	0.7361	0.618	0.0066

Table 2: Results in test set grouped by corpus domain.

The scores in the validation set (Table 3) follow a similar behaviour as those in the test set. For both, the corpus where our model achieves the best Pearson correlation is Biomed. However, looking at other metrics such as MAE, this corpus has the greatest error, with Europarl having the lowest. The differences may be because, even though the model may well capture the trend of the outputs, it could be more difficult to predict values in a corpus with higher variance of complexity scores, as is the case for Biomed (Figure 2).

Corpus	Pearson	Spearman	MAE	MSE
Bible	0.7353	0.6441	0.068	0.0072
Europarl	0.7946	0.7640	0.050	0.0039
Biomed	0.8571	0.8367	0.066	0.0075
Overall	0.8228	0.7643	0.062	0.0062

Table 3: Results in validation set grouped by corpus domain.

## 6 Ablation Study

Table 4 shows the contribution of each set of features (including predictions of fine-tuned models) to the final score. Although the predictions of

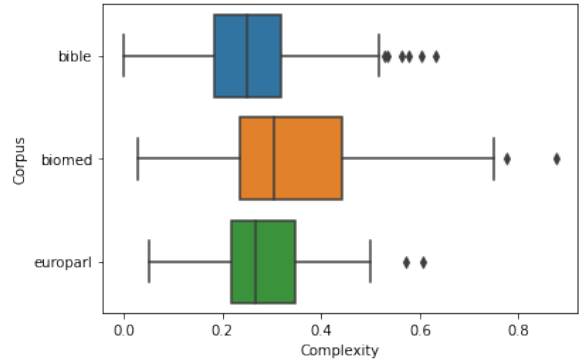


Figure 2: Distribution of the word complexity in validation set.

the fine-tuned Transformers-based models perform very well independently, the combination of all the predictions and the additional traditional features achieves the best performance in the validation set.

Another way of visualising the importance of each feature is using SHAP values (Lundberg and Lee, 2017). Figure 3 reports the 10 most important features for the LightGBM model, i.e. the impact of each feature in predicting the target complexity score. The X-axis shows the increase or decrease of target complexity, while the red and blue colours refer to the feature value’s size. For example, in the case of feature `size_of_sentence`, if the number of characters is larger there will be a positive impact, i.e. the complexity will increase. On the other hand, if the sentence length is smaller, there will be a negative impact, i.e. the complexity will decrease. We can observe that the most important feature is the predictions given by the BERT Multitask model since they have the greatest impact. This signals that WSD data could benefit predicting lexical complexity. It is also noted that the predictions of the Transformers-based models are in the top 5 of importance. Other features, such as the size of the sentence or the number of word senses, also have good contributions to the impact.

## 7 Conclusion

In this paper, we presented our system for the single word complexity prediction sub-task in the LCP Shared Task. Our approach consisted of combining lexical features and predictions from fine-tuned pre-trained Transformer-based models. We found that each set of features achieved a good performance on their own, and that combining all of them achieved our best result. In particular, we found that fine-tuning a pre-trained Transformer-

Approach	Pearson	Spearman	MAE	MSE
(a) BERT multitask with a window of 1	0.7972	0.7457	0.0642	0.00691
(b) BERT with a window of 1	0.7936	0.7507	0.0650	0.00703
(c) RoBERTa with a window of 1	0.7760	0.6946	0.0691	0.00776
(d) RoBERTa with a window of 2	0.7902	0.7179	0.0659	0.00729
(e) XLNet with a window of 1	0.7761	0.7253	0.0704	0.00795
(f) LightGBM with additional features	0.7859	0.7326	0.0663	0.0073
(a), (c), (d), (e) and (f)	0.8228	0.7643	0.0616	0.00618

Table 4: Results of each approach on validation data

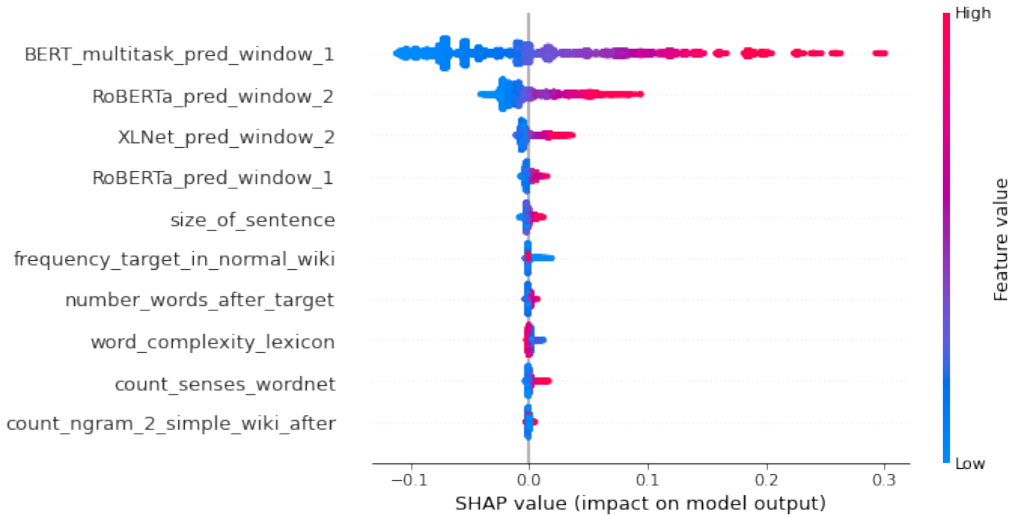


Figure 3: Shap analysis for the top 10 most important features

based model using multi-task learning with data from word sense disambiguation helped the most with learning to predict lexical complexity.

Considering that there were unseen tokens in validation and test sets, the task resembles a zero shot classification problem. Therefore, as future work, semi-supervised learning approaches or data augmentation algorithms could be explored, and training in a multitask fashion another transformer-based models like RoBERTa.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Pierre Finnimore, Elisabeth Fritsch, Daniel King, Alison Sneyd, Aneeq Ur Rehman, Fernando Alva-Manchego, and Andreas Vlachos. 2019. [Strong baselines for complex word identification across multiple languages](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 970–977, Minneapolis, Minnesota. Association for Computational Linguistics.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children’s books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.
- David Kauchak. 2013. [Improving text simplification language modeling using unsimplified text data](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria. Association for Computational Linguistics.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 3149–3157, Red Hook, NY, USA. Curran Associates Inc.



- Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.
- Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.
- Mounica Maddela and Wei Xu. 2018. A word-complexity lexicon and a neural readability ranking model for lexical simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3749–3760, Brussels, Belgium. Association for Computational Linguistics.
- Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.
- Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1).
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

# Uppsala NLP at SemEval-2021 Task 2: Multilingual Language Models for Fine-tuning and Feature Extraction in Word-in-Context Disambiguation

Huiling You, Xingran Zhu, and Sara Stymne

Uppsala University

Uppsala, Sweden

{Huiling.You.7480, Xingran.Zhu.2781}@student.uu.se

sara.stymne@lingfil.uu.se

## Abstract

We describe the Uppsala NLP submission to SemEval-2021 Task 2 on multilingual and cross-lingual word-in-context disambiguation. We explore the usefulness of three pre-trained multilingual language models, XLM-RoBERTa (XLMR), Multilingual BERT (mBERT) and multilingual distilled BERT (mDistilBERT). We compare these three models in two setups, fine-tuning and as feature extractors. In the second case we also experiment with using dependency-based information. We find that fine-tuning is better than feature extraction. XLMR performs better than mBERT in the cross-lingual setting both with fine-tuning and feature extraction, whereas these two models give a similar performance in the multilingual setting. mDistilBERT performs poorly with fine-tuning but gives similar results to the other models when used as a feature extractor. We submitted our two best systems, fine-tuned with XLMR and mBERT.

## 1 Introduction

SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC) (Martelli et al., 2021) is an extension from WiC (Pilehvar and Camacho-Collados, 2019), a shared task at the IJCAI-19 SemDeep workshop (SemDeep-5). WiC was proposed as a benchmark to evaluate context-sensitive word representations. The WiC dataset<sup>1</sup> consists of a list of English sentence-pairs. Each sentence-pair has a target word, and the task is to determine whether the target word is used in the same meaning or different meanings in the two sentences, thus as a binary classification task. MCL-WiC extends WiC to multilingual and cross-lingual datasets,<sup>2</sup> and covers 5

<sup>1</sup><https://pilehvar.github.io/wic/>.

<sup>2</sup><https://github.com/SapienzaNLP/mcl-wic>

Example	Label
The cat chases after the <i>mouse</i> . Click the right <i>mouse</i> button.	F
The cat chases after the <i>mouse</i> . La <i>souris</i> mange le fromage. (‘The <i>mouse</i> eats the cheese’)	T

Table 1: Examples for multilingual (top) and cross-lingual (bottom) word-in-context disambiguation.

languages: Arabic, Chinese, English, French, and Russian. The MCL-WiC task is also framed as a binary classification task: given a sentence-pair with a target word, either in the same language or in different languages, the goal is to determine whether the target word is used in the same meaning or in different meanings. Table 1 shows two example sentence pairs where the target word (*mouse*) has either an ‘animal’ or a ‘computer’ sense. In the **multilingual** setting, the two sentences are from the same language. In the **cross-lingual** setting, the two sentences are from different languages, English and one of the other four languages. Training data is only available for English–English, effectively leading to a zero-shot setting for the other languages.

Our main interest is to investigate the usefulness of pre-trained multilingual language models (LMs) in this MCL-WiC task, without resorting to sense inventories, dictionaries, or other resources. As our main method, we **fine-tune** the language models with a *span classification head*. We also experiment with using the multilingual language models as **feature extractors**, extracting contextual embeddings for the target word. In this setting, we also add information about syntactical dependency (i.e. head words and dependent words), with the intuition that it can contain relevant contextual information for disambiguation, as in Figure 1,

where the head words *chases* and *button* could help in disambiguating *mouse*. We compare three different LMs: XLM-RoBERTa (XLMR), multilingual BERT (mBERT) and multilingual distilled BERT (mDistilBERT).

We show that the fine-tuned models are stronger than any of the models based on feature extraction, by a large margin. XLMR is stronger than mBERT in the cross-lingual setting both with fine-tuning and feature extraction. mDistilBERT gives poor results with fine-tuning, but is competitive to the other LMs when used for feature extraction. Adding dependency syntax to our feature extraction method led to mixed results. We submitted our two strongest systems to the shared task, those fine-tuned with XLMR and mBERT.

## 2 Related Work

In WiC at SemDeep-5, many participating systems capitalized on contextualized word representations. The LMMS (Language Modelling Makes Sense) system by [Loureiro and Jorge \(2019\)](#) used word embeddings from BERT, together with sense embeddings from WordNet 3.0 ([Marciniak, 2020](#)). [Ansell et al. \(2019\)](#) used the contextualized representations from ELMo ([Peters et al., 2018](#)) and trained a separate classification model. [Soler et al. \(2019\)](#) experimented with several contextualized representations and used cosine similarity to measure word similarities. [Wang et al. \(2019\)](#) included WiC as one of the tasks in the proposed SuperGLUE benchmark, with the approach of fine-tuning BERT. At the end of the WiC evaluation period, the best result was achieved by [Wang et al. \(2019\)](#) with an accuracy of 68.36%, while human-level performance is 80%, as provided by the dataset curators.

[Scarlini et al. \(2020\)](#) recently proposed SensEmBERT<sup>3</sup>, a knowledge-based approach to sense embeddings for multiple languages. An important source for building SenseEmBERT is the contextualized representations from a pretrained language model. They experimented with SensEmBERT on both English and multilingual word sense disambiguation (WSD) tasks, and showed that SensEmBERT is able to achieve state-of-the-art result on both English and multilingual WSD datasets.

<sup>3</sup><http://sensembert.org/>

## 3 Multilingual Language Models

### 3.1 XLMR

XLMR (XLM-RoBERTa) is a scaled cross-lingual sentence encoder ([Conneau et al., 2020](#)), which is trained on 2.5T of data obtained from Common Crawl that covers more than 100 languages. XLMR has achieved state-of-the-art results on various cross-lingual NLP tasks.

### 3.2 mBERT

mBERT (multilingual BERT) is pre-trained on the largest Wikipedias ([Libovický et al., 2019](#)). It is a multilingual extension of BERT ([Devlin et al., 2019](#)) that provides word and sentence representations for 104 languages, which has been shown to be capable of clustering polysemic words into distinct sense regions in the embedding space ([Wiedemann et al., 2019](#)).

### 3.3 mDistilBERT

mDistilBERT (multilingual distilled BERT) is a light Transformer trained by distilling mBERT ([Sanh et al., 2019](#)), which reduces the number of parameters in mBERT by 40%, increases the speed by 60%, and retains over 97% of mBERT’s performance.

### 3.4 Sub-word models

XLMR, mBERT, and mDistilBERT all use sub-word models ([Wu et al., 2016](#); [Kudo and Richardson, 2018](#)), so the target word is usually represented by several sub-tokens. For example, given “qualify” as target word, it will be represented by “quali” and “fy” in XLMR. mBERT and mDistilBERT use a WordPiece model with a vocabulary size of 119,447 and XLMR use a SentencePiece model with a vocabulary size of 250,002. In our work, when the target word is represented by multiple sub-words, we use the averaged embedding as feature vector for the target word.<sup>4</sup>

## 4 System Description

We use the pre-trained language models in two different ways: for fine-tuning (Section 4.1) and as feature extractors (Section 4.2 - 4.3). Depending on whether feature transformation is involved, the features extracted can be further categorized into target

<sup>4</sup>We also explored summing sub-words, which gave similar results to averaging.

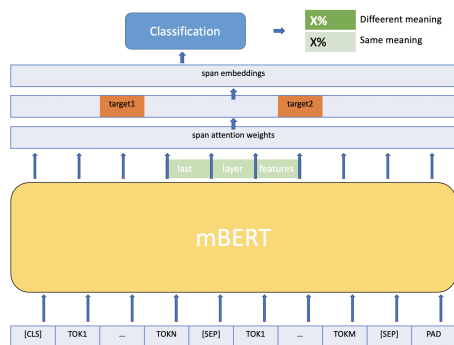


Figure 1: Model Structure of Fine-tuning mBERT

word embeddings (Section 4.2) and dependency-based syntax-incorporated word embeddings (Section 4.3). In the following sub-sections, we describe the three systems respectively. Due to time constraints we did not use XLMR in the systems with feature extraction.

#### 4.1 Fine-Tuning

The fine-tuning setup follows the architecture designed by Wang et al. (2019),<sup>5</sup> but extends to datasets in multiple languages. A *span classification head* is stacked on top of pre-trained language models, and attends only to the target words. The *span classification head* consists of a *span attention extractor* and a classifier. The *span attention extractor* is responsible for extracting the span embeddings, namely the target words embeddings. First, the unnormalized attention score of each token of the input document is computed. Span attention scores are the normalized scores of all tokens inside the span. Given the attention distributions over spans, each span gets a weighted representation of the last-layer hidden states of either mBERT, mDistilBERT or XLMR.

In this task, only the two target word spans will be returned, by masking out the rest of input. The attended span embeddings are then passed to the classifier, a linear transformation layer, to produce the output logits, which have a dimension of two, since there are only two labels (True or False). Figure 1 exemplifies the model structure when fine-tuning mBERT. The same structure also applies to XLMR and mDistilBERT.

<sup>5</sup>The package for SuperGLUE tasks is available at <https://github.com/nyu-ml1/jiant>

#### 4.2 Target Words Embeddings

In this setup, the multilingual language models serve as pure feature extractors, to get target word embeddings from last-layer hidden states. The input sample of a sentence-pair will then be the concatenation of the pair of target word embeddings.

We feed the two sentences separately to the models, and concatenate the embeddings for the two target words.<sup>6</sup> The extracted feature vectors are then fed to a classifier to perform the binary classification task. We experimented with two classifiers, logistic regression (LR) and a multi-layer perceptron (MLP).

#### 4.3 Dependency-based Syntax-Incorporated Embeddings

In this setup we ran a limited number of experiments. Only four languages (English, French, Chinese, and Russian)<sup>7</sup> and two pre-trained language models (mBERT and mDistilBERT) are explored.

The reasoning behind using syntax information to improve WiC classification results is as following. Given a pair of sentences, where the first sentence is “The cat chases after the mouse”, and the second one is “Click the right mouse button”, the target word *mouse* has different head words: in the first sentence, the singular verb *chases* is the head word, whereas in the second sentence, the noun *button* is the head word. Since it is more natural for a real mouse (as a small rodent) to be *chased* by its predators than to be related to a *button*, while in contrast, it is more common for a computer mouse (as a hand-held pointing device) to have a *button* than to be *chased*, the head words therefore reveal information on different contexts of the target word. The same reasoning applies to dependent words as well.

First, each sentence is parsed using the spaCy dependency parser,<sup>8</sup> from which we extract the target word, its head word, and its dependent word(s). Next, the sentence is passed to mBERT or mDistilBERT, and the corresponding target word embedding, head word embedding, and dependent

<sup>6</sup>We also experimented with concatenating the two sentences before feeding it to the LM, which gave slightly better results in some experiments. For consistency among all experiments we do not report these results.

<sup>7</sup>The latest version of spaCy (3.0.0), which is the dependency parsing library used in this work, does not support dependency parsing for Arabic, thus we do not run experiments on Arabic in this setup.

<sup>8</sup><https://spacy.io/usage/linguistic-features#dependency-parse>

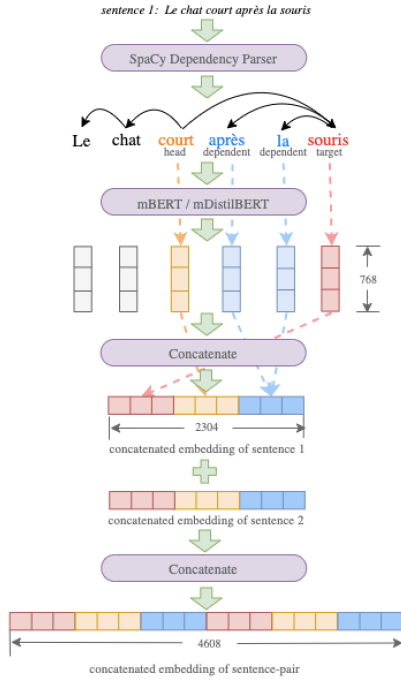


Figure 2: Construct a dependency-based syntax-incorporated embedding for a sentence-pair

word embedding(s) are retrieved, and concatenated. Note that if the target word has no head or dependent word, the *null* token embedding<sup>9</sup> is used instead; if the target word has more than one dependent word, all dependent word embeddings are summed element-wise.<sup>10</sup> Finally, the concatenated embeddings of two constituent sentences are further concatenated to form the sample feature vector of the sentence-pair, which is then fed to an MLP.

Figure 2 illustrates the process of constructing one such dependency-based syntax-incorporated embedding for a sentence-pair, of which the first sentence is *Le chat court après la souris*. The default embedding size of mBERT/mDistilBERT is 768. The sizes of different concatenated embeddings are shown in Figure 2. Again, we experimented with two classifiers, logistic regression and a multi-layer perceptron.

## 5 Experimental Setup

**Dataset** Only the datasets provided by SemEval-2021 Task 2 are used, see Table 2. All systems are trained on the English set, the multilingual development sets are used during development, and the

<sup>9</sup>That is, simply feeding the word *null* into mBERT/mDistilBERT and using the generated embedding directly.

<sup>10</sup>We also explored averaging the dependent word embeddings, which gave equivalent results to summing.

	Train	Dev	Test
en-en	8000	500	1000
ar-ar	–	500	1000
fr-fr	–	500	1000
ru-ru	–	500	1000
zh-zh	–	500	1000
en-ar	–	–	1000
en-fr	–	–	1000
en-ru	–	–	1000
en-zh	–	–	1000

Table 2: SemEval-2021 Task 2 Datasets. At development time, we only use half of the provided size (1000) of each dev set.

systems are tested on the multilingual and cross-lingual test sets.

**Fine-tuning** The three multilingual language models (mBERT, mDistilBERT, XLMR) are fine-tuned for three iterations, with batch size of 32, learning rate of 1e-5, and parameters optimized with AdamW (Loshchilov and Hutter, 2018), provided by Huggingface’s Transformers library<sup>11</sup>.

**Logistic Regression** All logistic regression (referred to as “LR” in the following sections) models are trained for 150 iterations, with batch size of 32, learning rate of 0.0025 and parameters optimized with standard stochastic gradient descent (SGD).

**MLP** All MLP models are 2-layer and follow the architecture suggested by Du et al. (2019), outputting classification label based on the probability:

$$\mathbf{p} = \text{softmax}(L_2(\text{ReLU}(L_1(\mathbf{e})))) \quad (1)$$

where  $\mathbf{e}$  is in the input embedding,  $L_i(\mathbf{x}) = \mathbf{W}_i\mathbf{x} + \mathbf{b}_i$  are fully-connected layers,  $\mathbf{W}_1 \in \mathbb{R}^{H \times H}$  and  $\mathbf{W}_2 \in \mathbb{R}^{2 \times H}$  are layer parameter matrices, and  $H$  is the input embedding size. All MLP models are trained for maximum 200 iterations, with learning rate of 0.001 and parameters optimized with Adam ( $\beta_1 = 0.9, \beta_2 = 0.999$ ) (Kingma and Ba, 2015).

**Language Model** We use the base version of all multilingual language models, with 12 layers, 12 attention heads, and hidden dimension of 768. Due to time constraints we did not use XLMR in the systems with feature extraction and an MLP.

<sup>11</sup><https://huggingface.co/transformers/>

	System	en-en	zh-zh	fr-fr	ru-ru	ar-ar	en-zh	en-fr	en-ru	en-ar
Fine-tune	XLMR	<b>84.5%</b>	<b>78.3%</b>	76.7%	73.1%	75.1%	<b>66.3%</b>	<b>70.9%</b>	<b>73.6%</b>	<b>65.2%</b>
	mBERT	82.9%	76.2%	<b>80.3%</b>	<b>73.6%</b>	<b>75.6%</b>	62.2%	66.3%	63.1%	59.4%
	mDistilBERT	75.5%	68.0%	66.8%	64.8%	68.9%	51.8%	53.4%	51.9%	50.9%
Feature Extractor	XLMR + LR	53.9%	55.4%	54.8%	57.2%	53.0%	58.2%	55.8%	55.4%	54.7%
	mBERT + LR	53.4%	53.5%	49.7%	51.7%	53.1%	52.0%	52.8%	52.8%	51.1%
	mDistilBERT + LR	55.7%	50.5%	52.6%	52.5%	51.9%	54.0%	52.5%	52.0%	51.6%
	mBERT + MLP	67.7%	51.4%	57.6%	54.2%	54.0%	47.4%	62.6%	55.6%	53.2%
	mDistilBERT + MLP	<b>66.6%</b>	59.1%	59.8%	61.8%	<b>56.0%</b>	48.2%	63.2%	57.4%	52.3%
	mBERT + Syntax + MLP	61.4%	52.7%	57.6%	57.0%	–	53.4%	57.8%	55.6%	–
	mDistilBERT + Syntax + MLP	67.0%	56.6%	58.2%	57.6%	–	54.0%	57.2%	56.2%	–

Table 3: System results on test sets. At task evaluation time, two fine-tuned systems were submitted, mBERT and XLMR; other systems were tested at post-evaluation time.

## 6 Results and Analysis

The evaluation results on the test sets are shown in Table 3. We can see that the fine-tuning approach is preferable to the feature extraction approach. All feature extraction variants fall behind the fine-tuned systems by a large margin. In many cases the systems based on feature extraction is just over chance performance (50%), and in a few cases it is even below it.

Among the fine-tuned systems, XLMR and mBERT give the best results, whereas mDistilBERT falls behind by quite a large margin in most cases, in several cases by more than 10 percentage points. The performance of mDistilBERT is especially weak in the cross-lingual setting. XLMR gives the best results for all cross-lingual language pairs, with an improvement over mBERT of 4.1–10.5 percentage points. The improvement is largest for English–Russian. For the multilingual setting, the difference between mBERT and XLMR is smaller with at most 3.6 percentage points. XLMR gives the best score in two cases and mBERT in three cases.

Among the systems with feature extraction, the relative performance of the three sets of contextual embeddings differ from the fine-tuning. Here, mDistilBERT are competitive to the other two embeddings. We only use XLMR with LR, and again, we see that it gives the best performance in the cross-lingual setting among all systems with LR, just as with fine-tuning. In the multilingual setting, XLMR is also strong, having the best result for three out of five languages. Compared to fine-tuning, mDistilBERT performs surprisingly well here. It is on par or better than mBERT in most cases across all settings.

Comparing the different architectures used with the feature extraction strategy, we see that using an MLP is preferable to LR, leading to large improvements in most cases. An exception is

English–Chinese, where the MLP without syntax performs worse than LR. For English–French on the other hand, the MLP outperforms LR by around 10 percentage points, whereas we see small improvements for English–Russian. Finally, the addition of syntax leads to mixed results. For the English–Chinese system, we see large improvements, whereas we see the opposite for English–French. For English–Russian as well as for all multilingual systems, the differences are overall smaller.

We also note that the performance is stronger for English–English than for the other languages in most settings. This is expected, since we only have English–English training data. A notable exception is for LR, where English–English performs considerably worse than in all other settings and is on par with the other languages in the same setting. With fine-tuning we overall see stronger results in the multilingual setting, than in the cross-lingual setting, where we mix language pairs. We do not see this difference for our feature extraction systems, however.

## 7 Conclusion and Future Work

We have investigated the use of three large language models for multilingual and cross-lingual word-in-context disambiguation. We found that fine-tuning the language models is preferable to using them as feature extractors either for an MLP or for logistic regression. Trying to add dependency-based syntax information in the MLP gave mixed results. We also found that XLMR performed better than mBERT in the cross-lingual setting, both with fine-tuning and feature extraction, whereas the two models had a more similar performance in the multilingual setting. mDistilBERT did not perform well with fine-tuning, but was competitive to the other models in the feature extraction setting. We submitted our two best systems, fine-tuning with

XLMR and mBERT to the shared task.

The fact that XLMR performs better than mBERT in the cross-lingual setting seems to indicate that it has a better representation of words across languages than mBERT and mDistilBERT. We think it would be worth investigating this hypothesis in more detail. XLMR and mBERT also use different sub-word models and another research direction is to explore the impact of this difference. We would also like to investigate the effect of using representations from different layers of the pre-trained multilingual language models.

## References

- Alan Ansell, Felipe Bravo-Marquez, and Bernhard Pfahringer. 2019. [An ELMo-inspired approach to SemDeep-5’s word-in-context task](#). In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 21–25, Macau, China. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. [Using BERT for word sense disambiguation](#). arXiv:1909.08358 [cs.CL].
- Diederik Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *Proceedings of the 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA*.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.
- Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. 2019. [How language-neutral is multilingual bert?](#) arXiv:1911.03310 [cs.CL].
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, Canada.
- Daniel Loureiro and Alípio Jorge. 2019. [Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.
- Jacek Marciniak. 2020. [Wordnet as a backbone of domain and application conceptualizations in systems with multimodal data](#). In *Proceedings of the LREC 2020 Workshop on Multimodal Wordnets (MMW2020)*, pages 25–32, Marseille, France. The European Language Resources Association (ELRA).
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. [WiC: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *Proceedings of The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, Vancouver, Canada.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. SenseBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *AAAI*, pages 8758–8765.
- Aina Garí Soler, Marianna Apidianaki, and Alexandre Allauzen. 2019. Limsi-multisem at the ijcai semdeep-5 wic challenge: Context representations for word usage similarity estimation. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 6–11.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.

Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings. In *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019): Long Papers*, pages 161–170, Erlangen, Germany.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.



# SkoltechNLP at SemEval-2021 Task 2: Generating Cross-Lingual Training Data for the Word-in-Context Task

Anton Razzhigaev<sup>1</sup>, Nikolay Arefyev<sup>2,3,4</sup>, and Alexander Panchenko<sup>1</sup>

<sup>1</sup>Skolkovo Institute of Science and Technology, Russia

<sup>2</sup>Samsung Research Center Russia, Russia

<sup>3</sup>Lomonosov Moscow State University, Russia

<sup>4</sup>HSE University, Russia

{anton.razzhigaev,a.panchenko}@skoltech.ru

narefjev@cs.msu.ru

## Abstract

In this paper, we present a system for the solution of the cross-lingual and multilingual word-in-context disambiguation task. Task organizers provided monolingual data in several languages, but no cross-lingual training data were available. To address the lack of the officially provided cross-lingual training data, we decided to generate such data ourselves. We describe a simple yet effective approach based on machine translation and back translation of the lexical units to the original language used in the context of this shared task. In our experiments, we used a neural system based on the XLM-R (Conneau et al., 2020), a pre-trained transformer-based masked language model, as a baseline. We show the effectiveness of the proposed approach as it allows to substantially improve the performance of this strong neural baseline model. In addition, in this study, we present multiple types of the XLM-R based classifier, experimenting with various ways of mixing information from the first and second occurrences of the target word in two samples.

## 1 Introduction

The goal of the second task of SemEval-2021 (Martelli et al., 2021) is to perform multilingual and cross-lingual word-in-context disambiguation. More specifically, participants are asked to distinguish whether the meanings of a target word in two provided contexts are the same or not. Organizers provided a training set of 8 000 English language (en-en) context pairs and validation sets of 1 000 context pairs for English-English (en-en), French-French (fr-fr), Russian-Russian (ru-ru), Arabic-Arabic (ar-ar), and Chinese-Chinese (zh-zh) languages. Since no cross-lingual training data were provided, except for a very small trial set barely usable for training, we decided to venture into generating such data automatically.

Essentially, the given task is a binary classification problem. The first question was which supervised model to use for the classification of context pairs. Recently, pre-trained masked language models such as BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) have been used to reach promising results in a variety of similar NLU classification tasks. Thus, we decided to make use of contextualized embeddings from XLM-R, which provides multilingual-lingual embeddings for more than 100 languages, covering all language pairs of interest in the shared task. In all our experiments, this model is used as the backbone.

A straightforward way of solving tasks where two contexts are to be compared, as the word-in-context tasks, is to use deep contextualized embeddings and train a classifier over these embeddings as has been explored in the original monolingual word-in-context task (Pilehvar and Camacho-Collados, 2019). Note that commonly embeddings of two contexts are simply concatenated (Ma et al., 2019) and this operation is asymmetric. In our work, we explored various symmetric ways of aggregating embeddings from two contexts.

The contributions of our work are two-fold. First, we present a simple yet effective method for the generation of cross-lingual training data, showing that it can substantially improve the performance compared to the model trained using monolingual data. Second, we test various ways of encoding two input target word occurrences contexts using the XLM-R model.

## 2 Baseline Supervised WiC System

Massively multilingual transformers pretrained with language modeling objectives XLM-R were shown to be useful for zero-shot cross-lingual transfer in NLP (Lauscher et al., 2020). As a baseline, we rely on a supervised system that takes as an

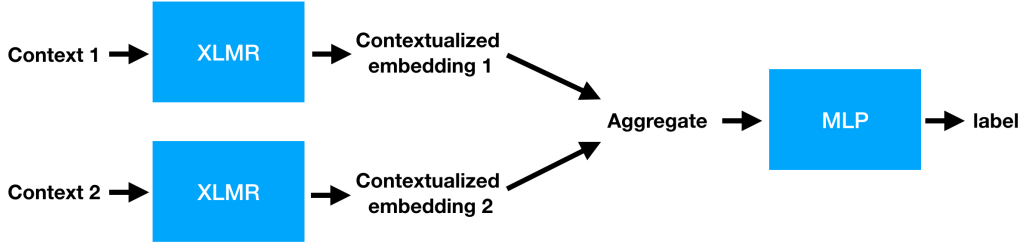


Figure 1: Principal scheme of the supervised model used in our experiments. Context pairs are sent to XLM-R base and then contextualized embeddings of target words are merged and sent to MLP which outputs the prediction probability of them having the same meaning. XLM-R is frozen.

input two sentences and spans corresponding to occurrences of the target word. Pre-trained multilingual encoders are used to represent sentences in different languages in the same space.

## 2.1 Multilingual Sentence Encoder

We use XLM-R masked language model (Conneau et al., 2020) as the basis in our experiments as it supports all required languages by the shared task. This is a multilingual transformer-based masked language model pre-trained on a large corpus consisting of texts from the Web in 100 languages. This model is a strong baseline on various NLU tasks. Besides, our preliminary experiments have shown that it is capable of encoding sentences written in different languages in the same vector space. This property, therefore, is crucial as it allows similar methods, which were used to successfully solve the monolingual word-in-context task in the past (Pilehvar and Camacho-Collados, 2019).

Figure 1 presents the overall schema of the model used in our experiments. The XLM-R model is used for obtaining contextualized embeddings of the target words, while a multi-layered perceptron is used to perform the classification. We thoroughly tested various meta-parameters of this architecture. Different aggregation methods are presented in the following section.

## 2.2 Symmetric Aggregation of Deep Contextualized Embeddings

Each training example consists of two contexts with marked target words and a label representing these words being in the same or different meanings. In our approach, both contexts are sent to XLM-R, and then contextualized embeddings for target words (averaged activations from two last layers) are extracted and merged into one embed-

ding with the following symmetric procedure: concatenate element-wise product of two embeddings and the absolute value of the element-wise difference of two embeddings. This helps to obtain a vector containing deep contextualized representation of a target word in both contexts. Then this merged embedding is sent to a 3-layer MLP which outputs the probability of two words been in the same senses (Figure 1).

More specifically, we test different ways of aggregating embeddings from two contexts. We conducted several experiments, including two asymmetric aggregation approaches and four symmetric. Let  $\vec{a} = \{a_1, \dots, a_n\}$  be the contextualized embedding of a target word from the first context and  $\vec{b} = \{b_1, \dots, b_n\}$  – from the second.

The tested two following commonly used asymmetric approaches of merging two embeddings:

1. Concatenating of embeddings:

$$\vec{c} = \{a_1, \dots, a_n, b_1, \dots, b_n\}$$

2. Difference of embeddings:

$$\vec{c} = \{a_1 - b_1, \dots, a_n - b_n\}$$

Besides, we tested four symmetric approaches to embedding aggregation listed below:

1. Sum of embeddings:

$$\vec{c} = \{a_1 + b_1, \dots, a_n + b_n\}$$

2. Elementwise product of embeddings:

$$\vec{c} = \{a_1 \cdot b_1, \dots, a_n \cdot b_n\}$$

3. Absolute value of difference of embeddings:

$$\vec{c} = \{|a_1 - b_1|, \dots, |a_n - b_n|\}$$

4. Concatenation of variants 2 and 3:

$$\vec{c} = \{a_1 \cdot b_1, \dots, a_n \cdot b_n, |a_1 - b_1|, \dots, |a_n - b_n|\}$$

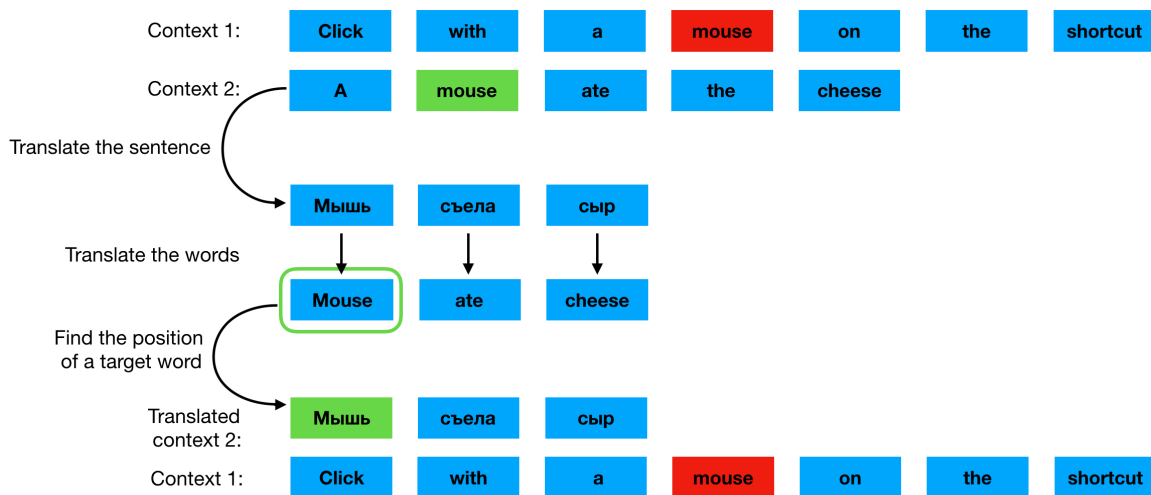


Figure 2: An illustration of the cross-lingual data generation. Given two sentences, we pick one and translate it to a target language. Then in order to find the position of a target word, every lexical unit is back-translated and compared with a target lemma. If the target word is found, the translated sentence is used in addition to the second sentence from the initial pair as a new cross-lingual training example.

### 3 Generation of Cross-lingual Training Data using Machine Translation

In this section, we describe a machine-translation-based method for the generation of synthetic training data for the cross-lingual word-in-context task to address the lack of cross-lingual training data usable for the supervised model described above.

#### 3.1 Method

We suggest the forward-backward translation approach, which helps not just to translate a sentence but to identify the position of a target word which is essential for the word-in-context task.

We decided to use the provided 8 000 English-English pairs of texts and translate them to the desired languages. But there is a difficulty: after translation the position of target word in the context is unknown, or even target word is replaced by several words like in the following example of Russian-English translation (the target words are underlined):

- ru: “налей кипяток в стакан”
- en: “pour boiling water into a glass”

In our experiments, we filter similar examples which do not have a uniword translation of a target word.

Overall, our algorithm amounts to the following procedure:

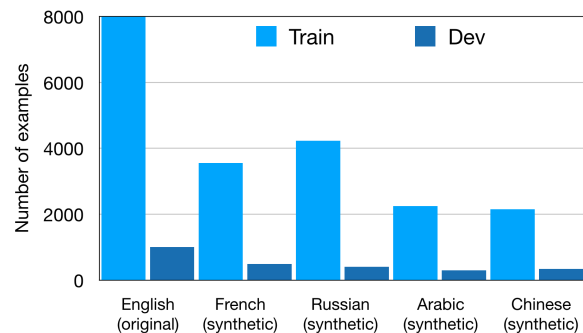


Figure 3: Amount of the English training/development data and amount of synthetic cross-lingual data generated from it.

1. Translate a sentence from the source language to a target with a neural machine translation.<sup>1</sup>
2. Back translate every word independently without a context. For the translation of single words, we use the word2word<sup>2</sup> library
3. If there is a target lemma in the list of back-translated words, then the lemma index in the back-translated words list is the index of the target word in the translated sentence.
4. If there is no target lemma in the list of back-translated words, then we do not use this sentence.

<sup>1</sup><https://github.com/ssut/py-googletrans> (Google translate Python API)

<sup>2</sup><https://github.com/kakaobrain/word2word>

**label:** False  
**text1:** Her parents allegedly *received* threats from the police to make them drop the charges against the soldiers.  
**text2:** Специальный докладчик *получил* информацию о положении в тюрьмах из государственных и негосударственных источников.

**label:** False  
**text1:** Similarly, in the *event* of a transplant, the deceased donor must continue to be entitled to respect for his body.  
**text2:** Следует прилагать усилия для разработки программ обмена и даже спортивных *мероприятий*, таких как футбольные матчи.

**label:** True  
**text1:** The Committee recommends such a *procedure* be reflected in the provisions contained in the proposed draft protocol.  
**text2:** Его *процедуры* незамедлительных действий также заслуживают высокой оценки.

**label:** True  
**text1:** During a summer the two students started doing their own research on a *roulette* wheel which they had bought.  
**text2:** Используя MACS для записи вращений стола *рулетки* с течением времени, компьютер может предсказывать будущие результаты.

Figure 4: Examples of generated synthetic cross-lingual data.

Method	Type	fr-fr
Concatenating	A	67.0 $\pm$ 2.0
Difference	A	65.4 $\pm$ 1.3
Summation	S	79.6 $\pm$ 1.3
Elementwise product	S	81.8 $\pm$ 1.4
Absolute difference	S	81.5 $\pm$ 1.8
Concat of symmetric	S	<b>82.1 <math>\pm</math> 1.4</b>

Table 1: Symmetric (S) vs asymmetric (A) ways of merging XLMR-large contextualized embeddings. Concatenation of symmetric: concatenation of elementwise multiplication and absolute difference.

A schematic illustration of our algorithm is presented in Figure 2.

### 3.2 Generation Result

The synthetic examples for English-Russian are presented in Figure 4. The first two sentence pairs (with the False, F label) represent negative training examples, i.e., pairs of sentences in which target words are used in different senses (across languages). The last two sentence pairs (with the True, T label) represent contexts where words are used in the same sense. As one may observe, the generated examples are semantically coherent, and the position of the target word was identified correctly using our back-translation heuristic.

The overall amount of generated training cross-lingual examples for each language compared to the amount of initial English language data presented in Figure 3. The unequal number of samples is due to the translation errors and the fact that back translation does not always point to the original word. That is why we also present results for the fixed sizes of synthetic datasets for each language in the Table 2.

## 4 Experiments and Results

Below we report the results of the two setups of this shared task: multi- and cross-lingual settings. We train the model six times; reporting mean and standard deviation of accuracy on the test dataset.

### 4.1 Results on Various Embedding Aggregation Methods

All embedding aggregation methods were tested on the French language development set, been trained on the English training set. The experimental results are presented in Table 1. Experimental results demonstrate that the suggested symmetric aggregation of embeddings is a better choice for such symmetric problems like two context comparisons than a common asymmetric aggregation. We suppose that this experimental fact is caused by the symmetric nature of a comparison problem and hence all similar tasks should exploit symmetrically merged embeddings.

### 4.2 Results on Multilingual Datasets

In a multilingual setting, context pairs are provided in four languages, but pairs are written in the same language. As XLM-R provides contextualized text representations in the same space for different languages, we supposed that our XLM-R based model should work in a zero-shot setting: being trained on only one language shows decent results on other languages. To verify our hypothesis, we conducted the following experiments:

1. Training only on 8 000 MCL-WiC English context pairs (zero-short setting).
2. Training on 8 000 MCL-WiC English context pairs (from the training set) + 5 000 multi-language pairs (from development set).

The results are presented in Table 2: substantially higher results than the random baseline (50

Training set	en-en	fr-fr	ru-ru	ar-ar	zh-zh
English train data	87.5 ± 0.9	82.1 ± 1.4	78.9 ± 1.7	69.2 ± 1.9	65.2 ± 1.5
English train and multilingual dev	<b>89.9 ± 0.8</b>	<b>84.1 ± 1.5</b>	<b>86.5 ± 1.1</b>	<b>72.4 ± 1.3</b>	<b>70.2 ± 1.0</b>

Table 2: Results on test data in multi-lingual setting.

Training set	en-fr	en-ru	en-ar	en-zh
English train data	64.1 ± 2.7	61.4 ± 2.1	59.1 ± 2.1	52.9 ± 1.3
English train and multilingual dev	66.5 ± 1.2	62.0 ± 1.0	58.9 ± 1.8	52.1 ± 0.7
Synthetic for each language (fixed)	70.1 ± 2.1	69.7 ± 1.9	<b>63.1 ± 2.1</b>	<b>60.1 ± 1.4</b>
Synthetic for each language (full)	72.6 ± 2.0	71.4 ± 1.4	62.7 ± 2.1	<b>60.1 ± 1.4</b>
All data	<b>73.5 ± 1.9</b>	<b>72.8 ± 1.5</b>	58.8 ± 1.7	52.1 ± 0.6

Table 3: Results on test data in cross-lingual setting.

percent) are obtained. Note that in this case, the dataset is balanced, so the most frequent class classifier is equivalent to the random one. This confirms the fact that a zero-shot transfer using XLM-R is possible.

### 4.3 Results on Cross-lingual Datasets

In a cross-lingual setting, context pairs are provided in four languages, and pairs are written in different languages. The main challenge of the task is cross-lingual Word-in-Context disambiguation. We approach this task from two sides: zero-shot learning capabilities of multilingual XLM-R based systems and generation with a machine translation of cross-lingual synthetic training data. To verify that zero-shot learning works in a cross-lingual setting and synthetically generated data improves the results in cross-lingual tests, we performed the following experiments:

1. Training only on 8 000 MCL-WiC English context pairs (zero-shot setting).
2. Training on 8 000 MCL-WiC English context pairs + 10 000 multi-language pairs.
3. Training on synthetic cross-lingual examples. Training and testing each language separately.
4. Training on all data including MCL-WiC train, development sets, and synthetic cross-lingual data for all languages simultaneously.

Results are presented in the Table 3. The best results for Russian and French are obtained using all the available data, including the generated synthetic dataset. For Arabic and Chinese, the best results

are obtained using synthetic data only. Overall, performance in all settings for Chinese and Arabic is substantially lower. This may be due to the more complex morphological structure of these languages and the way how the XLM-R pre-trained model handles it (while the European languages like French and Russian have similar alphabet structures). Overall, the experiments suggest the usefulness of the generated synthetic data for the solution of the cross-lingual word-in-context task.

## 5 Conclusion

In this paper, we presented a solution to the cross-lingual word-in-context task. The main challenge of this task, as formulated by the organizers, is the lack of explicit training data. To address it, we developed a way of generating synthetic cross-lingual data for the word-in-context disambiguating task; we demonstrate the positive influence of such synthetic data on the performance of a model on test datasets.

As the baseline model in our experiments, a supervised model based on XLM-R pre-trained language model (Conneau et al., 2020) was used. We performed tests of various settings based on this model and demonstrated that symmetric aggregation of embeddings for context comparison tasks outperforms asymmetric ways on zero-shot and supervised settings.

The code and the produced data, enabling reproducing our experiment, are available online.<sup>3</sup>

<sup>3</sup><https://github.com/skoltech-nlp/cross-lingual-wic>

## References

- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Anne Lauscher, Vinit Ravishankar, Ivan Vulić, and Goran Glavaš. 2020. [From zero to hero: On the limitations of zero-shot language transfer with multilingual Transformers](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4483–4499, Online. Association for Computational Linguistics.
- Xiaofei Ma, Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. 2019. [Universal text representation from BERT: an empirical study](#). *CoRR*, abs/1910.07973.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. [WiC: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

# Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning

**Boris Zhestiankin**

Moscow Institute of Physics and Technology  
Moscow, Russia

`zhestyankin.ba@phystech.edu`

**Maria Ponomareva**

ABBYY  
HSE University  
Moscow, Russia

`maria.ponomareva@abbyy.com`

## Abstract

This paper presents our contribution to SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). Our experiments cover English (EN-EN) sub-track from the multilingual setting of the task. We experiment with several pre-trained language models and investigate an impact of different top-layers on fine-tuning. We find the combination of Cosine Similarity and ReLU activation leading to the most effective fine-tuning procedure. Our best model results in accuracy 92.7%, which is the fourth-best score in EN-EN sub-track.

## 1 Introduction

The increasing progress in Natural Language Processing is closely related with development of word representations. The context-independent word embeddings, such as word2vec (Mikolov et al., 2013) and fastText (Bojanowski et al., 2017) brought the idea of measuring the relatedness of the meanings as the distance between the vectors encoding them. The introduction of the methods of pre-training context dependent embeddings, such as ELMo (Peters et al., 2018), ULMFit (Howard and Ruder, 2018), and BERT (Devlin et al., 2018) made the next crucial breakthrough overcoming the shortcomings of previous methods to encode the meaning. Despite the fact that the primal objective of word embeddings is to encode the meaning of words, it is not obvious how to evaluate them directly. While common manner to examine the superiority of particular type of embeddings is to look at their performance on some downstream tasks, the more direct way to evaluate their ability to represent semantic is challenging.

SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC) (Martelli et al., 2021) presents a new framework to evaluate embeddings. In this paper we

present our contribution for the task. We explore the potential of different pre-trained context-dependent embeddings based on pre-trained language models. We find that the Cosine Similarity can produce fruitful results when used for fine-tuning the weights of the pre-trained models, while adding linear layers to learn the similarity from the limited data leads to instant overfitting.

## 2 Background

The traditional approach to evaluate the ability of embeddings to catch the meaning of words is Word Sense Disambiguation (WSD) task (Navigli, 2009). WSD is defined as classification problem, when a given word is classified between its predefined senses. WSD by design comes with an important limitation, being connected directly with predefined sense inventories such as WordNet<sup>1</sup> (Fellbaum, 2005).

The Word in Context (WiC) benchmark (Pilehvar and Camacho-Collados, 2019) addresses these limitations. The task proposes a binary classification setting for English, when, given two sentences  $s_i$  and  $s_k$  and two words  $w_i$  and  $w_k$  in them, the system needs to decide whether the word  $w_i$  in  $s_i$  and  $w_k$  in  $s_k$  have same or different meanings. The main advantage of WiC task is a possibility to expand its consideration to the languages that lack such sense inventories.

MCL-WiC extends the WiC approach to new senses and new languages, covering data in five languages: Arabic, Chinese, English, French and Russian. The task provides data of two types: in the multilingual setting one needs to predict the label to the pair of sentences in one language (AR-AR, ZH-ZH, EN-EN, FR-FR, RU-RU sub-tracks), in the cross-lingual setting the first sentence is in English and the second one is in one of the four

<sup>1</sup><https://wordnet.princeton.edu>

other considered languages (EN-AR, EN-ZH, EN-FR, EN-RU sub-tracks).

After preliminary experiments we decided to focus our efforts on the only sub-track with training data, namely the English sub-track from the multilingual setting. Our solution<sup>2</sup> is fourth placed in the EN-EN leaderboard with 92.7% accuracy and is 0.6% behind the winner.

### 3 System overview

Approaching the task we conduct multiple experiments with a variety of architectures, however all of them are deeply based on contextual embeddings fine-tuning. For our experiments we use pre-trained embeddings from BERT and XLM-RoBERTa (Conneau et al., 2020) models and fine-tune them for our task.

#### 3.1 Target word embeddings

Design of BERT and XLM-RoBERTa models assumes that text is first split to tokens and embeddings for these tokens are evaluated. Therefore we define our technique to obtain the embeddings, representing target words in the sentences.

For a single sentence we take embeddings of all sub-tokens corresponding to the target word in it and max pool them into one embedding. Repeating this procedure for both sentences in each pair we obtain two embeddings as the result: first — corresponding to target word in the first sentence and second — corresponding to target word in the second sentence.

#### 3.2 Multilayer Perceptron Architecture

In our initial setup we build a system based on Multilayer Perceptron neural network. The purpose of this approach is to train the system to predict that target words have the same meaning in both sentences.

This model calculates embeddings of the target word in both sentences of the pair and concatenates them together, taking the result as an input layer. The model contains one hidden layer with 100 neurons, ReLU activation before it and an output layer, activated by sigmoid.

Interpreting the model output as the probability that target words have the same meaning in both of the sentences, we predict True if the output turns

---

<sup>2</sup>Source code, experiments, requirements and results can be found at <https://github.com/zhestyatsky/MCL-wiC>

out to be greater than 0.5 or we predict False otherwise.

To enrich the knowledge of the model about the task we also experiment with a slightly different input, making use of [CLS] tokens. Each [CLS] token represents the whole sentence. Taking [CLS] tokens embeddings for each sentence in a pair we concatenate them together and afterwards concatenate the result with an input layer (consisting of target word embeddings concatenation) defined above. We use the resulting embedding as an input layer for our model and do not change other parameters in the setup.

#### 3.3 Cosine Similarity Architecture

As an alternative to Multilayer Perceptron approach we define a Cosine Similarity approach, illustrated on Figure 1. which proves to be our best system for the task. The purpose of this approach is to train the system to predict the probability that the target word has the same meaning in both sentences.

During training our system takes embeddings of the target word in each sentence in a pair and calculates Cosine Similarity between them. It activates the similarity through ReLU layer. The result value is considered the output of the model.

After the training is finished we have to make predictions, which is achieved by defining the probability threshold as a hyperparameter. In this way we predict True if the output of the model is greater than the threshold or False otherwise.

To maximize the accuracy of the model we calculate the probability threshold by building the Receiver Operating Characteristic (ROC) curve and choosing the value corresponding to the maximum difference between true positive and false positive rates.

We note that in this approach no new weights are introduced in contradistinction to Multilayer Perceptron approach. Therefore only pre-trained weights of BERT and XLM-RoBERTa models are fine-tuned.

To provide a comparison option for Cosine Similarity approach we also try applying sigmoid as an activation instead of ReLU.

#### 3.4 Datasets

Speaking about the datasets for training and validation we fully utilize train and development English data provided by the competition organisers for the EN-EN sub-track. However, to achieve the



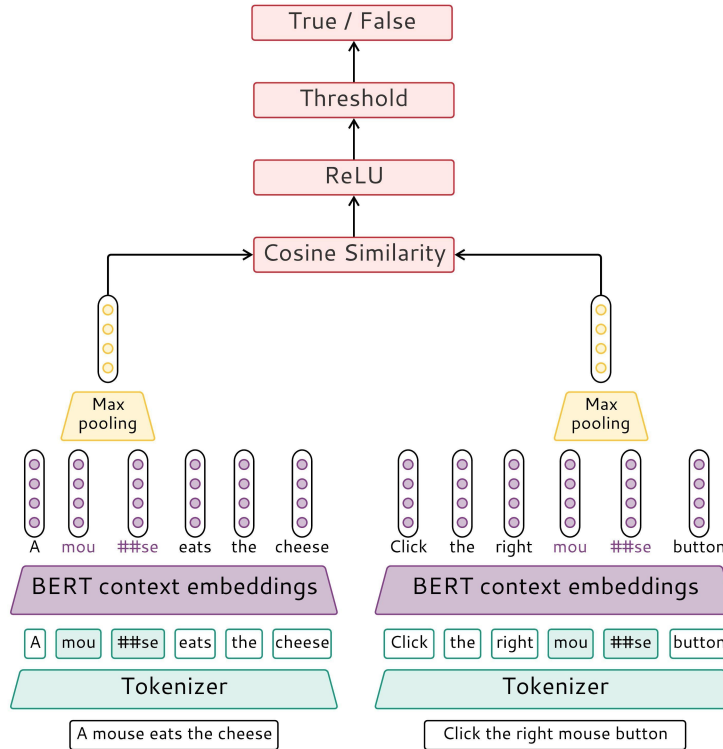


Figure 1: The scheme presents Cosine Similarity Architecture, which was used in the model achieving the best performance in our experiments.

best possible results we extend our train and development datasets with WiC dataset (Pilehvar and Camacho-Collados, 2019)<sup>3</sup>, included to SuperGLUE (Wang et al., 2019) benchmark, for English sentence pairs.

We also conduct an experiment with our best model, using only default datasets provided by competition organisers. This experiment will be described at the end of Results section.

#### 4 Experimental setup

In our setup we mix train and development data and split it randomly by unique lemmas in proportion 97.5% to 2.5%. Having 14680 samples in the first chunk and 386 samples in the second chunk we use the first chunk for training and the second for validation.

During training, the data is processed by batches of size 8. Each sentence is split into 118 tokens maximum. In this way it is guaranteed that the longest sentence in the dataset is not going to be truncated.

Experiments with four different types of embeddings are conducted<sup>4</sup>:

- *bert-base-cased*: 12-layer, 768-hidden, 12-heads, 109M parameters;
- *bert-large-cased*: 24-layer, 1024-hidden, 16-heads, 335M parameters;
- *xlm-roberta-base*: ~270M parameters with 12-layers, 768-hidden-state, 3072 feed-forward hidden-state, 8-heads;
- *xlm-roberta-large*: ~550M parameters with 24-layers, 1024-hidden-state, 4096 feed-forward hidden-state, 16-heads.

We train our models for a maximum of 8 epochs and define an early stopping criteria. Every half of epoch (after training on the half of all the batches) we check if the loss on validation dataset is decreasing. If the loss does not decrease for 2 checks in a row, we stop training.

In all our experiments we use Binary Cross Entropy Loss as the loss function and AdamW optimizer with a learning rate set to 1e-5.

To conduct experiments we use version 1.7.1 of PyTorch (Paszke et al., 2019) together with version 0.8.2 of torchvision<sup>5</sup> and version 0.8.1 of torchtext<sup>6</sup>,

<sup>3</sup><https://pilehvar.github.io/wic/>

<sup>4</sup><https://huggingface.co/transformers>

<sup>5</sup><https://github.com/pytorch/vision>

<sup>6</sup><https://github.com/pytorch/text>

version 1.1.6 of PyTorch Lightning<sup>7</sup> framework and version 4.2.2 of HuggingFace’s Transformers (Wolf et al., 2020). From the latter we obtain BERT and XLM-RoBERTa model implementations.

As we define a probability threshold as a hyperparameter in Cosine Similarity approach, we provide its values for all experimental configurations in the Table 1.

embeddings	activation	threshold
<i>xlm-roberta-large</i>	<b>sigmoid</b>	0.680
<i>xlm-roberta-base</i>		0.632
<i>bert-large-cased</i>		0.609
<i>bert-base-cased</i>		0.678
<i>xlm-roberta-large</i>	<b>ReLU</b>	0.638
<i>xlm-roberta-base</i>		0.642
<i>bert-large-cased</i>		0.519
<i>bert-base-cased</i>		0.509

Table 1: Probability thresholds for Cosine Similarity Architecture. Abbreviations used: **activation** stands for *activation function* used, **threshold** stands for *probability threshold* of the model.

## 5 Results

In the Table 2 the results of the fine-tuning of language models with Multilayer Perceptron on top are presented. During the experiments we found out that for this dataset not only additional linear layers can not learn to measure the distance effectively, but they lead to overfitting in a few epochs. It is seen by the number of the passed epochs before the early stopping.

As [CLS] token is designed to accumulate sentence meaning we expected it to make the representations for each instance in a pair more complete. The results in the Table 2 show that the usage of [CLS] tokens give a moderate improvement to all models except for one with *xlm-roberta-large* embeddings.

Pre-trained language models, like BERT and XLM-RoBERTa, have the property of associating close vectors with similar words. Therefore to provide a baseline for the model described in Cosine Similarity approach we measure the accuracy of it without additional fine-tuning. Due to the technique used to evaluate the probability thresholds, the accuracies for configurations with different activations are identical in this case. Accuracies for dif-

<sup>7</sup><https://github.com/PyTorchLightning/pytorch-lightning>

embed	add cls	epochs	val	test
<b>XLMR-l</b>	<b>yes</b>	2.5	0.585	0.579
<b>XLMR-b</b>		2.5	0.580	0.580
<b>BERT-l</b>		3.5	0.585	0.548
<b>BERT-b</b>		2.5	0.588	0.565
<b>XLMR-l</b>	<b>no</b>	2	0.484	0.519
<b>XLMR-b</b>		2.5	0.590	0.611
<b>BERT-l</b>		3	0.598	0.583
<b>BERT-b</b>		2.5	<b>0.601</b>	<b>0.592</b>

Table 2: Accuracy of models with Multilayer Perceptron Architecture. Abbreviations used: **embed** stands for *embeddings*, **add cls** defines if [CLS] token embedding was used, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

ferent embeddings and thresholds for sigmoid and ReLU activations can be found in Table 3. Viewing the results on validation dataset we can estimate the quality of the approach and the results on test dataset confirm its relevance. Best accuracy on validation dataset is provided by *bert-large-cased* embeddings. In addition, the thresholds in Table 3 show how differently the vector spaces are arranged for BERT and XLM-RoBERTa models: for the second, a threshold of about 0.99 distinguishes vectors of words with different meanings from words with the same meanings.

embed	sigm thld	ReLU thld	val	test
<b>XLMR-l</b>	0.73	0.995	0.645	0.659
<b>XLMR-b</b>	0.72	0.994	0.666	0.719
<b>BERT-l</b>	0.66	0.64	<b>0.710</b>	<b>0.780</b>
<b>BERT-b</b>	0.69	0.77	0.690	0.780

Table 3: Accuracy of models with Cosine Similarity Architecture without fine-tuning. Abbreviations used: **embed** stands for *embeddings*, **sigm thld** stands for *probability threshold of model using sigmoid activation*, **ReLU thld** stands for *probability threshold of model using ReLU activation*, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. As models are not fine-tuned, accuracies on validation and test datasets are independent of the activation function. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

Finally, Table 4 presents results of the experimental setup when the language models are fine-tuned using Cosine Similarity measure. It is worth mentioning that in such a setup there are no additional weights and only the layers of the language model are changing. It can be seen that such an architecture allows the model not to overfit for longer epochs.

embed	activ	epochs	val	test
<b>XLMR-l</b>	<b>sigm</b>	6	0.661	0.748
<b>XLMR-b</b>		3	0.679	0.746
<b>BERT-l</b>		3	0.728	0.823
<b>BERT-b</b>		3	0.676	0.727
<b>XLMR-l</b>	<b>ReLU</b>	5.5	0.785	0.876
<b>XLMR-b</b>		2	0.730	0.769
<b>BERT-l</b>		<b>4.5</b>	<b>0.808</b>	<b>0.927</b>
<b>BERT-b</b>		4	0.790	0.889

Table 4: Accuracy of models with Cosine Similarity Architecture. Abbreviations used: **embed** stands for *embeddings*, **activ** stands for the *activation function* used, **sigm** stands for *sigmoid activation function*, **val** stands for *accuracy on validation dataset*, **test** stands for *accuracy on test dataset*. We refer to *xlm-roberta-large* as **XLMR-l**, to *xlm-roberta-base* as **XLMR-b**, to *bert-large-cased* as **BERT-l** and to *bert-base-cased* as **BERT-b**.

While conducting the experiments, we judged the models by their performance on the validation dataset, not being able to check how representative it is. According to the obtained scores, the validation dataset is representative enough and is more challenging for the models than the test dataset.

To provide a convenient report we conduct an experiment with our best model (using *bert-large-cased* embeddings together with Cosine Similarity Architecture, using ReLU activation), which only uses data provided by organisers. We perform no further processing with the data and use it as is: train dataset is used for training and development dataset for validation. Being trained for 4 epochs the model in the experiment demonstrates 0.886 accuracy on validation dataset and **0.913** accuracy on test dataset. This result shows that using additional data leads to better performance.

## 6 Error analysis

Our best model leads to accuracy 92.7%. It means that our model has erroneously labeled 73 sentences in the 1000-sentence testset. The error analysis revealed that our model is not biased towards

one or another class, it produced 37 false negative predictions and 36 false positive predictions. The next observation is related to the construction feature of the dataset. The dataset is organized in the following manner: for each combination of lemma and POS-tag there are two instances in the dataset. All three possible combinations of labels are presented, with prevalent case when one pair is labeled False and second True. The peculiarity of the dataset is that both instances have the same first sentence. We found that 20 out of 73 errors have these repeating first sentence. In other words, if the model produces incorrect prediction for one instance for lemma it tends to make a mistake for the second instance in the dataset. Due to the described peculiarity of the data, we can not speculate that certain lemma is a stumbling block for the model or it is just a context of the first sentence, that for example differs by genre or thematically from second sentence and complicates the prediction. The manual analysis of the errors has not revealed instances that could be considered hard and unclear for human assessment.

In order to reveal objectively hard instances among the errors of the best model, we have intersected the mislabeled pairs for all the models fine-tuned with Cosine Similarity. The intersection indicated that all but two instances were predicted correctly by at least one of the models. We can conclude that no objectively hard instances were presented in the erroneously labeled pairs by the best model. Additionally, the possible conclusion could be that an ensemble of our models could result in even more powerful solution for the task.

## 7 Conclusion

We have provided an overview of different approaches to fine-tune pre-trained language models for the task that is naturally suitable for them – detecting the distance between representations of the words.

We have showed that, for the data of given amount and type, learning distance between words in context with Multilayer Perceptron neural network is not applicable and generally leads to overfitting.

Using Cosine Similarity to predict probability during pre-trained embeddings fine-tuning leads to much more promising results, when activated with ReLU layer.

## References

- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). Cite arxiv:1810.04805Comment: 13 pages.
- Christiane Fellbaum. 2005. [WordNet and wordnets](#). In *Encyclopedia of Language and Linguistics*, pages 665–670, Oxford. Elsevier.
- Jeremy Howard and Sebastian Ruder. 2018. [Fine-tuned language models for text classification](#). *CoRR*, abs/1801.06146.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Distributed representations of words and phrases and their compositionality](#). In *Neural and Information Processing System (NIPS)*.
- Roberto Navigli. 2009. Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS*, 41(2):1–69.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. [WiC: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

# SzegedAI at SemEval-2021 Task 2: Zero-shot Approach for Multilingual and Cross-lingual Word-in-Context Disambiguation

Gábor Berend

Institute of Informatics

University of Szeged

Árpád tér 2., Szeged, Hungary

berendg@inf.u-szeged.hu

## Abstract

In this paper, we introduce our system that we participated with at the multilingual and cross-lingual word-in-context disambiguation SemEval 2021 shared task. In our experiments, we investigated the possibility of using an all-words fine-grained word sense disambiguation system trained purely on sense-annotated data in English and draw predictions on the semantic equivalence of words in context based on the similarity of the ranked lists of the (English) WordNet synsets returned for the target words decisions had to be made for. We overcame the multi,-and cross-lingual aspects of the shared task by applying a multilingual transformer for encoding the texts written in either Arabic, English, French, Russian and Chinese. While our results lag behind top scoring submissions, it has the benefit that it not only provides a binary prediction whether two words in their context have the same meaning, but also provides a more tangible output in the form of a ranked list of (English) WordNet synsets irrespective of the language of the input texts. As our framework is designed to be as generic as possible, it can be applied as a baseline for basically any language (supported by the multilingual transformed architecture employed) even in the absence of any additional form of language specific training data.

## 1 Introduction

A major obstacle in solving word sense disambiguation (WSD) problems in a supervised manner is the scarcity of annotated training corpora. As the construction of high quality sense-annotated training data can be extremely labor-intensive and difficult (Gale et al., 1992), the Word-in-Context (WiC) disambiguation task was recently proposed by Pilehvar and Camacho-Collados (2019) as a surrogate for the traditional WSD problem. While in the traditional fine-grained WSD setting, the aim is to

assign a precise and often nuanced meaning to a word in its context according to some sense inventory, WiC is framed as a binary classification problem, where the task is to decide whether two target words originating from a pair of input sentences have the same meaning. This kind of binary decision can also be made in the absence of a nuanced sense inventory, making the annotation process less demanding and also more suitable across languages (Raganato et al., 2020).

In this paper, we analyze the utilization of multilingual transformer-based language models for performing both multi-lingual and cross-lingual WiC in the zero-shot setting, by employing nothing but English sense annotated training data and utilizing the model predictions in a transductive model that is capable of performing zero-shot WSD and WiC disambiguation for any language that is supported by the multilingual transformer encoder model that gets employed.

Loureiro and Jorge (2019) showed that a simple, nearest neighbor approach relying on contextual word embeddings can achieve impressive WSD results in English. In our follow-up work (Berend, 2020), we demonstrated, how sparse contextualized word representations can be exploited for obtaining significant improvements over the LMMS approach introduced by Loureiro and Jorge (2019). Our shared task participation was focused on comparing the two techniques in a zero-shot multilingual and cross-lingual WiC evaluation setting.

## 2 System overview

At the core of our multi,-and cross-lingual WiC systems, we employed fine-grained WSD systems, originally intended to solely handle English texts. The two models that we employed were the LMMS (Loureiro and Jorge, 2019) and the S-LMMS (Berend, 2020) approaches. We dub the

latter solution as S-LMMS, highlighting its resemblance to the LMMS approach and the fact that it operates with *sparse* contextualized word representations. Both LMMS and S-LMMS requires sense-labeled training data for constructing their respective fine-grained WSD models.

We provide a brief overview of the two approaches and encourage readers interested in more details to read the original papers (Loureiro and Jorge, 2019; Berend, 2020) introducing them. LMMS and S-LMMS both has in common, that they encode the inputs with a transformer model (BERT-large). LMMS constructs a prototype vector for each English synset based on the BERT-encoded vectors of the sense-annotated training data and the actual contents of the English WordNet glosses. For a given token in its context, LMMS takes its BERT-encoded contextualized vector and finds the nearest synset prototype for determining its sense.

The way S-LMMS differs from LMMS is that it additionally incorporates a sparsity inducing dictionary learning step, which turns the contextualized word representations into a sparse format, i.e., to such vectors that contain a high fraction ( $> 90\%$ ) of zero coefficients. Additionally, the methodology for creating the synset prototype vectors has substantial differences between the two approaches, as LMMS uses the actual contextualized embeddings pertaining to a certain synset as prototypes, whereas S-LMMS distills a vectorial representation to each synset based on an information theoretic measure.

The important technical change that we performed over the previously described fine-grained WSD models, so that they can be employed in the cross-lingual setting, is that we replaced the BERT-large encoders that the LMMS and S-LMMS models use by default to the XLM-RoBERTa-large (Conneau et al., 2020) architecture. We shall refer to the variants of LMMS and S-LMMS that were obtained by relying on XLM-RoBERTa as an encoder as opposed to BERT-large as mLMMS and mS-LMMS, owing to the multilingual nature of XLM-RoBERTa. We used the transformers library (Wolf et al., 2020) for obtaining the contextualized multilingual embeddings for our experiments.

When performing fine-grained WSD in English, one can simply restrict the scope of predicting the most likely synset for some word to those that are deemed viable for a given word in WordNet. Addi-

tionally, one can also filter the synsets over which the prediction is performed, based on the part-of-speech category of a word in question. With these heuristics, it is possible to reduce the number of synsets that a word can belong to a few dozens of synsets even for the most ambiguous cases.

In order to test a solution that is as generic as possible, we did not integrate any of these heuristics into our framework, meaning that our models returned a ranked list over *all* the 117,659 English WordNet synsets to any word from some sentence. This way, our solution can also work basically any language (supported by the multilingual transformer employed), even in the absence of a multilingual sense-inventory resource such as BabelNet (Navigli and Ponzetto, 2010) and also when we have no access to the part-of-speech information, nor to a part-of-speech tagger for some language. These design choices ensures that we are able to handle a much wider range of languages as if we decided otherwise. To this end, we regard our approach a particularly good fit being used as a baseline for WSD related evaluations involving low-resource languages.

As mentioned previously, our  $\ast$ LMMS models assigned a ranked list of 117,659 English synsets to every target word irrespective of the language of the sentence it was written in. Since the ranking of the synsets for a given word was performed over all the synsets of WordNet, it would be too restrictive to expect that words with identical meaning should be assigned the exact same most likely English synset. To this end, we measured the similarity for a pair of ranked lists that a model returned for a pair of words in their contexts and decided about the semantic equivalence of the two words based on that similarity score. As the similarity scores calculated for the ranked lists of synsets that fit those pairs of words that have the same meaning are expected to be higher on average, we decided to determine a threshold for the similarity scores of the ranked lists above which we predicted the two words to have the same meaning, and to have a different meaning otherwise.

We experimented with three strategies for measuring the similarity of two ranked synset lists for a pair of words. Let  $S_1$  and  $S_2$  refer to the ranked lists of WordNet synsets assigned to two words. As the bottom of the ranking is arguably not as meaningful as its top-ranked elements, we decided to formulate  $S_1^{(100)}$  and  $S_2^{(100)}$ . These ranked lists

differed from  $S_1$  and  $S_2$  in that they contained their top 100-ranked elements, respectively.<sup>1</sup>

Since we only focus on the highest ranked synsets from  $S_1$  and  $S_2$ , it is almost sure that certain element from  $S_1^{(100)}$  are not included in  $S_2^{(100)}$ , and vice versa. As such, the usage of standard rank correlation scores would be inconvenient for measuring the similarity between ranked lists  $S_1^{(100)}$  and  $S_2^{(100)}$ . One motivation behind the introduction of ranking-biased overlap (RBO) (Webber et al., 2010) was particularly this, i.e. to provide such a distance metric that is capable of operating between non-conjoint rankings. RBO is an overlap-based metric, that can operate over such rankings when the ranked elements themselves are not totally identical. To this end one of our metric for measuring the similarity between  $S_1^{(100)}$  and  $S_2^{(100)}$  was based on the RBO metric.

Our other approach for measuring the similarity of ranked lists  $S_1^{(100)}$  and  $S_2^{(100)}$  was to simply take their Jaccard similarity, i.e. the fraction of the size of their intersection and the elements in their union. As a third approach, we calculated the harmonic mean of the mean reciprocal rank (MRR) of the highest ranked synset from  $S_1^{(100)}$  in the ranked list  $S_2^{(100)}$  and similarly, that of the highest ranked synset from  $S_2^{(100)}$  in  $S_1^{(100)}$ . We then based our predictions with the similarity scores calculated by either of the above manner.

Instead of using some supervised approach, we determined a threshold for the similarity score for a pair of ranked synset lists  $S_1^{(100)}$  and  $S_2^{(100)}$ , above which we predicted that the words they got assigned to had identical meaning. We determined this threshold in a transductive manner, without using any of the labeled training or development set sentence pairs at all. For the cross-lingual evaluation it would have been impossible at the first place, as no annotated pairs of sentences were released during the shared task.

We used expectation maximization for determining the similarity threshold above which we predicted a pair of words to have the same meaning. That is, we took all the similarity scores that we calculated for a certain test set based on the  $S_1^{(100)}$  and  $S_2^{(100)}$  ranked synset lists, and fitted a Gaussian Mixture Model over the similarity scores. That way, we managed to fit a Gaussian distribution for

<sup>1</sup>Experiments with different thresholds (10, 25, 50, 250 and 500) also provided similar results that we omit for brevity.

the similarity scores of pairs of words with identical and different meanings. We identified the fitted Gaussian distribution with the higher expected value to be the one that corresponds to the distribution of similarity scores for those words that have identical meaning. As expectation maximization algorithms are prone to find local optima, we initialized each model 100 times and chose the one which resulted in the best log-likelihood score. Our decisions for a particular test sample was then based on the density functions on the similarity scores of the two classes determined by the best fitting model.

### 3 Experiments

We tested our approach on both the multilingual and the cross-lingual subtasks of the shared task (Martelli et al., 2021). The multilingual test sets consisted of sentence pairs that were written in the same language (either Arabic, English, French, Russian or Chinese), whereas, an input was comprised of an English and a non-English (either Arabic, French, Russian or Chinese) sentence for the cross-lingual scenario.

The fine-grained WSD model that we built our system on was trained over English sense-annotated training data. We used two sources of training signal, the SemCor dataset as well as the Princeton WordNet Gloss Corpus (WNGC), which has been shown to improve fine-grained WSD results (Vial et al., 2019; Berend, 2020). Unless stated otherwise, we used these three sources of sense-annotated training data for obtaining our \*LMMS models.<sup>2</sup>

#### 3.1 Monolingual all-words WSD experiments

We first evaluated LMMS and S-LMMS models on standard fine-grained all-words disambiguation data included in the unified evaluation framework from (Raganato et al., 2017). What we were interested here is the change in the standard WSD performance of these systems when replacing the English specific BERT-large model that LMMS and S-LMMS originally employ to XLM-RoBERTa-large. At this point we evaluated our fine-grained WSD performance in terms of F-score over the concatenation of the five standard evaluation benchmarks from SensEval2 (Edmonds and Cotton, 2001), SensEval3 (Mihalcea et al., 2004), SemEval 2007 Task 17 (Pradhan et al., 2007), SemEval 2013 Task 12

<sup>2</sup>Our source code can be found at [https://github.com/begab/sparsity\\_makes\\_sense](https://github.com/begab/sparsity_makes_sense)

Layer(s) used	LMMS	S-LMMS	Layer(s) used	mLMMS	mS-LMMS
21	0.758	<b>0.790</b>	21	0.702	<b>0.757</b>
22	0.763	<b>0.785</b>	22	0.692	<b>0.753</b>
23	0.760	<b>0.786</b>	23	0.679	<b>0.749</b>
24	0.745	<b>0.780</b>	24	0.648	<b>0.728</b>
21-24	0.757	<b>0.788</b>	21-24	0.692	<b>0.754</b>

(a) BERT-large

(b) Using XLM-RoBERTa-large

Table 1: Comparison of the model performances towards fine-grained WSD using the standard benchmark from (Raganato et al., 2017) (consisting of the concatenated test sets of the SensEval2-3 and the SemEval 2007, 2013 and 2015 shared tasks on fine-grained WSD), when using different layers from different transformer models and model variants \*LMMS.

(Navigli et al., 2013), SemEval 2015 Task 13 (Moro and Navigli, 2015). This test set consisted of 7,253 English test cases in total.

Table 1 includes our results using the four different models that were using different layers from the transformer model that was employed for encoding the input texts. As expected, replacing the English specific transformer model to a multilingual encoder resulted in a decreased performance, however, the overall decrease was not very severe. Comparison of the results in Table 1a and Table 1b reveals that the performance of S-LMMS is less affected by the integration of the multilingual RoBERTa model in place of the English-only BERT model for encoding. Additionally, using the encodings from the 21th layer of the transformer models seem to provide a slight edge over the utilization of the concatenation of the last four layers irrespective of the encoder and the specific WSD model used. To this end, we participated in the shared task-related with such \*LMMS models that were using the contextualized word representations from the 21th layer alone, as opposed to the average of the last four layers.

### 3.2 Evaluation on the shared task data

In Table 2, we list those test scores that we obtained by differently configured versions of our architecture. Our results span the different strategies for performing all-words fine-grained WSD (mLMMS/mS-LMMS) and different strategies for calculating the similarity between two ranked list of most likely synsets assigned to the test words (Jaccard/MRR/RBO) as described earlier in Section 2.

We can see from Table 2 the same phenomenon as for our monolingual fine-grained WSD evalua-

tions in Table 1, i.e., the mS-LMMS approach had a clear advantage over LMMS for both the multilingual and the cross-lingual evaluation settings.

Regarding the effects of choosing different ways to calculate the similarity scores between a pair of ranked lists of synsets, the application of the Jaccard similarity and the RBO metric-based similarity seems to perform very similarly, with the mean reciprocal rank based similarity scoring slightly underperforming the other two alternatives. Overall, the results seem to be balanced over the languages, with the choice of the fine-grained WSD system being more influential to the final results as the choice of the similarity calculation between the ranked lists of synsets returned by them to a pair of test words.

For training our \*LMMS models, we decided to experiment with the integration of a recent source of sense tagged training dataset, UWA (Loureiro and Camacho-Collados, 2020), which is a sense-annotated corpus containing unambiguous words from Wikipedia and OpenWebTex. We relied on the recommended version of the UWA corpus which contains 10 example sentences for each unambiguous word. By expanding the number of sense annotated training text, it becomes possible to increase the coverage of the fine-grained WSD systems. We investigated the downstream effects for our WiC system of extending the amount of sense annotated training data used by our fine-grained WSD systems.

Our evaluation results over the same set of models as in Table 2, with the only difference that we additionally used the UWA10 sense-annotated corpus for creating our all-words WSD models are included in Table 3. This additional training corpus was not always helpful, however, increased our



	Jaccard		MRR		RBO	
	mLMMS	mS-LMSS	mLMMS	mS-LMMS	mLMMS	mS-LMSS
ar	60.0	61.4	<b>62.1</b>	60.7	59.2	59.5
en	62.6	67.2	<b>70.6</b>	70.4	62.6	66.1
fr	62.1	66.6	62.4	60.9	60.7	<b>66.9</b>
ru	58.9	67.1	63.9	66.6	56.6	<b>67.3</b>
zh	55.9	63.8	56.0	63.8	56.7	<b>64.6</b>
avg.	59.9	<b>65.2</b>	63.0	64.5	59.2	64.9

(a) Multilingual results

en-*	Jaccard		MRR		RBO	
	mLMMS	mS-LMSS	mLMMS	mS-LMMS	mLMMS	mS-LMSS
ar	59.9	<b>66.3</b>	59.1	64.4	61.3	62.2
fr	61.2	63.9	59.5	63.1	59.6	<b>64.6</b>
ru	63.7	<b>66.4</b>	61.2	60.2	62.7	65.9
zh	64.2	65.3	51.5	65.6	62.9	<b>66.3</b>
avg.	62.3	<b>65.5</b>	57.8	63.3	61.6	64.8

(b) Cross-lingual results

Table 2: The effects of applying different similarity measures (Jaccard/MRR/RBO) to the different fine-grained WSD approaches (mLLS/mS-LMMS) integrated into our zero-shot multilingual and cross-lingual WiC framework.

average accuracy by a slight ( $\approx 1\%$ ) margin.

## 4 Conclusions

In this paper, we introduced our cross,-and multilingual WiC framework that we approached from an all-words fine-grained word sense disambiguation perspective. As such, our model not only provides a yes or no answer for a pair of words in their contexts, but also provides a more tangible explanation for it in the form of the similarity between the ranked lists of English WordNet synsets assigned to the target words.

During the design of our approach, we made such choices that would make our framework conveniently applicable to new languages without the need for any training data. Although the results of our framework lags behind the top performing systems, due to of its convenient applicability to new languages and the fact that practically no additional training data is required for applying it to new and possibly low-resourced languages, we think it can provide an easy to use baseline in further WiC-related research efforts.

## Acknowledgments

The research presented in this paper was supported by the Ministry of Innovation and the National Research, Development and Innovation Office within the framework of the Artificial Intelligence Na-

	Jaccard		MRR		RBO	
	mLMMS	mS-LMSS	mLMMS	mS-LMSS	mLMMS	mS-LMSS
ar	58.2	64.1	65.0	<b>66.9</b>	58.0	63.8
en	62.4	68.2	69.3	<b>70.2</b>	63.0	68.7
fr	62.4	67.8	60.1	61.6	62.8	<b>68.3</b>
ru	57.9	<b>68.7</b>	64.8	67.7	60.5	66.2
zh	51.3	63.0	<b>64.9</b>	63.9	53.3	64.2
avg.	58.4	<b>66.4</b>	64.8	66.1	59.5	66.2

(a) Multilingual results

en-*	Jaccard		MRR		RBO	
	mLMMS	mS-LMSS	mLMMS	mS-LMMS	mLMMS	mS-LMSS
ar	59.5	<b>65.8</b>	60.2	64.7	58.8	63.2
fr	60.6	<b>64.9</b>	62.0	59.8	60.5	64.5
ru	63.0	65.9	61.8	59.6	62.7	<b>68.3</b>
zh	60.3	66.0	53.1	65.5	60.9	<b>67.1</b>
avg.	60.9	65.7	59.3	62.4	60.7	<b>65.8</b>

(b) Cross-lingual results

Table 3: The effects of incorporating the UWA10 sense-annotated corpus during the training phrase of our fine-grained English WSD model that served as a basis of our WiC architecture.

tional Laboratory Programme. The author is grateful for the fruitful discussions with Tamás Szakálos whose research was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002. The project has been supported by the European Union and co-funded by the European Social Fund.

## References

- Gábor Berend. 2020. [Sparsity makes sense: Word sense disambiguation using sparse contextualized word representations](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8498–8508, Online. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Philip Edmonds and Scott Cotton. 2001. [SENSEVAL-2: Overview](#). In *The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems, SENSEVAL '01*, pages 1–5, Stroudsburg, PA, USA. Association for Computational Linguistics.
- William A. Gale, Kenneth W. Church, and David

- Yarowsky. 1992. [A method for disambiguating word senses in a large corpus](#). *Computers and the Humanities*, 26(5):415–439.
- Daniel Loureiro and Jose Camacho-Collados. 2020. [Don't neglect the obvious: On the role of unambiguous words in word sense disambiguation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3514–3520, Online. Association for Computational Linguistics.
- Daniel Loureiro and Alípio Jorge. 2019. [Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. [SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation \(MCL-WiC\)](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. [The SENSEVAL-3 english lexical sample task](#). In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28, Barcelona, Spain. Association for Computational Linguistics.
- Andrea Moro and Roberto Navigli. 2015. [SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking](#). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.
- Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. [SemEval-2013 task 12: Multilingual word sense disambiguation](#). In *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Roberto Navigli and Simone Paolo Ponzetto. 2010. [BabelNet: Building a very large multilingual semantic network](#). In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. [WiC: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.
- Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. [SemEval-2007 task 17: English lexical sample, srl and all words](#). In *Proceedings of the 4th International Workshop on Semantic Evaluations, SemEval '07*, pages 87–92, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. [Word sense disambiguation: A unified evaluation framework and empirical comparison](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.
- Alessandro Raganato, Tommaso Pasini, Jose Camacho-Collados, and Mohammad Taher Pilehvar. 2020. [XLWiC: A multilingual benchmark for evaluating semantic contextualization](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7193–7206, Online. Association for Computational Linguistics.
- Loïc Vial, Benjamin Lecouteux, and Didier Schwab. 2019. [Sense Vocabulary Compression through the Semantic Knowledge of WordNet for Neural Word Sense Disambiguation](#). In *Global Wordnet Conference*, Wroclaw, Poland.
- William Webber, Alistair Moffat, and Justin Zobel. 2010. [A similarity measure for indefinite rankings](#). *ACM Trans. Inf. Syst.*, 28(4).
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

# ReCAM@IITK at SemEval-2021 Task 4: BERT and ALBERT based Ensemble for Abstract Word Prediction

Abhishek Mittal      Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

amittal@iitk.ac.in

ashutoshm@cse.iitk.ac.in

## Abstract

This paper describes our system for Task 4 of SemEval-2021: Reading Comprehension of Abstract Meaning (ReCAM). We participated in all subtasks where the main goal was to predict an abstract word missing from a statement. We fine-tuned the pre-trained masked language models namely BERT and ALBERT and used an ensemble of these as our submitted system on Subtask 1 (ReCAM-Imperceptibility) and Subtask 2 (ReCAM-Nonspecificity). For Subtask 3 (ReCAM-Intersection), we submitted the ALBERT model as it gives the best results. We tried multiple approaches and found that Masked Language Modeling (MLM) based approach works the best.

## 1 Introduction

Computers' ability to understand, represent, and express text with abstract meaning is a fundamental problem towards achieving true natural language understanding. In past decades, significant advancement has been achieved in representation learning. SemEval-2021 Task 4 : Reading Comprehension of Abstract Meaning (ReCAM) (Zheng et al., 2021) explores the ability of machines to understand abstract concepts and proposes to predict abstract words just as humans do while writing article summaries. In the shared task, text passages are provided to read and understand abstract meaning. It consists of three subtasks where the first two subtasks are based on two different definitions of abstractness 1) Imperceptibility (Spreen and Schulz, 1966) and 2) Non-specificity (Changizi et al., 2008) and the third subtask discusses their intersection.

Many cloze-style reading comprehension datasets like CNN/Daily Mail (Hermann et al., 2015) and Children's Book Test (CBTest) dataset (Hill et al., 2016) and models (Dhingra et al., 2016;

Munkhdalai and Yu, 2017) similar to this task exist, where a missing word has to be inferred. However, these previous datasets and models have mostly focused on inferring concrete words or concepts like named entities, but this task moves the focus from concreteness to abstractness of words in reading comprehension. This can prove to be quite useful for current ongoing research in the field of abstractive summarization.

We participated in all the three subtasks. We mainly used an ensemble of BERT and ALBERT as our final model for submission on subtasks 1 and 2. We were ranked 13th on Subtask 1 and 11th on Subtask 2. We submitted the ALBERT model on Subtask 3. All of our code is made publicly available on Github<sup>1</sup>. We approached this task in two ways. One is a Multiple Choice Question answering (MCQ) based approach and other a Masked Language Modeling (MLM) approach. Through experiments, we concluded that such tasks are best addressed using a masked language model.

The rest of the paper is organised as follows. Section 2 describes the problem statement formally and also gives a brief description of the dataset provided by the task organizers. Section 3 introduces the related work. Section 4 describes our proposed approach and Section 5 gives the experimental details. We enlist our results in Section 6 with a brief error analysis. Finally, we give concluding remarks in Section 7.

## 2 Background

### 2.1 Problem Description

A passage  $P$ , a followup question  $Q$  with a **@placeholder** and a list of candidate answer words  $W = \{W_1, W_2, W_3, W_4, W_5\}$  are given as an input to the model. The task is to output the correct answer

<sup>1</sup>[https://github.com/amittal151/SemEval-2021-Task4\\_models](https://github.com/amittal151/SemEval-2021-Task4_models)

Sr No.	Ambiguous Examples					
1	<b>Article</b> : [7 January 2016 Last updated at 10:26 GMT Robin Sollis packs it all away in his garage - leaving no room for his car. His is one of 14 homes on a Cirencester street which have raised £1,200 for charity with their displays this year. Tracey Miller reports.] <b>Question</b> : Taking down this @placeholder Christmas lights display in Gloucestershire will take its owner several days.					
	<b>Options</b>	Impressive	Special	Annual	Fake	New
	There is nothing about Gloucestershire in the article !					
2	<b>Article</b> : [20 April 2017 Last updated at 00:28 BST The reason is that cafe and shop owners who provide wi-fi could be held liable for illegal activity that happens on their network, such as piracy or illegal downloads. But campaigners are trying to change this law. Videojournalist: Joe Miller. ] <b>Question</b> : For a country with a reputation as a tech powerhouse , public wi - fi is @placeholder difficult to find in Germany.					
	<b>Options</b> :	often	also	surprisingly	particularly	rather
	There are many plausible answers (rather is also feasible) to the question. Use of words may depend on personal choices.					

Figure 1: Few ambiguous examples in Subtask 1 Dev set (Option marked in green is the correct answer)

$W_i$  from  $W$  ( $W_i \in W$ ) by learning the function  $F$  such that  $W_i = F(P, Q, W)$

The first two subtasks focus on the two different definitions of abstractness and the third subtask captures the relationship between the two views of abstractness. The evaluation metric for all three subtasks is the accuracy of the predictions made by the model. The subtasks are enlisted below :

1. **ReCAM-Imperceptibility** - Abstract words refer to ideas and concepts that are not immediately perceivable by our senses like culture, objective, etc.
2. **ReCAM-Nonspecificity** - According to this definition, abstract words refer to holistic terms, e.g., animal, body, etc.
3. **ReCAM-Intersection** - In the third subtask, the system needs to be trained on one definition of abstractness (Imperceptibility) and evaluated on the other (Nonspecificity) and vice-versa.

Task	Training	Dev	Test
1	3227	837	2025
2	3318	851	2017

Table 1: Number of examples in dataset

## 2.2 Data Description

The task organizers have provided training and validation dataset for Subtask 1 and Subtask 2. Each training and validation set example is in the form of a dictionary containing an article, a question and 5 options. One word in the question is missing and is represented by “@placeholder”, and we have to predict the word out of the given 5 options.

The data set has English news articles and questions are constructed from the summaries of these articles. The data statistics are provided in table 1. The dataset poses two major challenges. Firstly, the passages are quite long. Their distribution is shown in figure 2 and 3. The long article length leads to a loss of context when we truncate the article in a transformer based model due to its max token length limits. Secondly, the dataset contains some ambiguous examples where more than one correct answer could be feasible or the question’s context is missing from the article. Some examples are shown in the Figure 1.

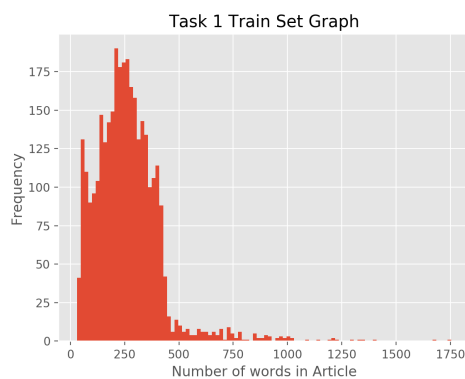


Figure 2: Task 1 Article statistics

## 3 Related Work

Much work has been done for the prediction of concrete words unlike ours where we need to predict abstract words in reading comprehensions. Gated Attention Reader (Dhingra et al., 2016) predicts missing concrete words in CNN/Dailymail datasets with a high accuracy. The attention mechanism plays a crucial role in recognizing which sections of the article are more important to answer the questions. Extracting context from the article is a vital

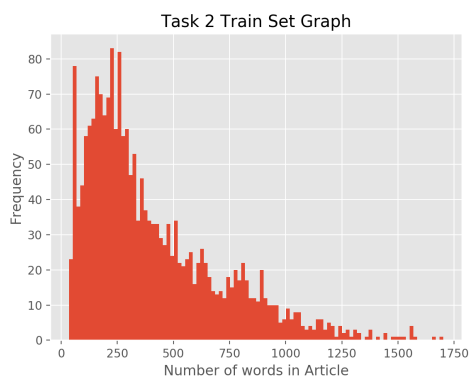


Figure 3: Task 2 Article statistics

part of the task. This task requires comprehensive natural language understanding, going beyond the meaning of individual words and sentences. We explored some of the pre-trained transformer models (Vaswani et al., 2017) as these capture the context better due to the self-attention mechanism. Moreover, pre-trained models are readily available.

The task is somewhat similar to a multiple choice question answering task. We experimented with the MCQ based approach as mentioned by Radford (2018) where a linear layer is built over the transformer and the correct answer is predicted by applying softmax over the probabilities of each option.

One approach to get context from the article is to extract the most relevant sentences to the question with sentence similarity techniques (Reimers and Gurevych, 2019). We experimented with this approach and extracted “Top-k” sentences that were most semantically similar to the given question from the article.

One of the major challenge in this task is to handle the long length of the article. Pappagari et al. (2019) discuss the approach of using hierarchical transformers for text classification problem to tackle long passages. BERT is applied to text segments and an LSTM layer or transformer is applied to get document embedding.

Another approach is to model the shared task as a masked language modeling task. The transformer based models like BERT (Devlin et al., 2018) and ALBERT (Lan et al., 2020) have been trained via the masked language modeling objective. BERT has also been trained on the Next Sentence Prediction task, and ALBERT has been trained on the Sentence Ordering task. In Lan et al. (2020), it is mentioned that Sentence Ordering task is a better

way to understand the similarity and extracting context from two sentences and thus, ALBERT works better than the BERT model.

## 4 System Overview

We explored multiple models and methodologies. We first experimented using an encoder with an attention based approach. From their results, we observed that an MLM based approach would work better than an MCQ based approach.

Consequently, we tried BERT and ALBERT models and their ensemble with the MLM based approach. However, for comparison purposes, we also worked with the MCQ method and its system is described below along with our other approaches.

### 4.1 Encoders with Attention

#### 4.1.1 Binary Classification with Attention

We tried a binary classification based approach where we give each option a score of being a correct answer. Our model consisted of two encoders, followed by a binary classifier. One encoder is for encoding the question and one for encoding the article. First, we feed the question into the question encoder which gives us the context vector of the question. Then, we feed the article along with the hidden weights from the question encoder into the article encoder and apply attention weights over them to find the context of question within the article. Finally, we input an option word, the hidden weights obtained from the article encoder and the context vector from question encoder into the binary classifier layer which gives us the score for the given option. The option word with highest score is predicted as our answer.

#### 4.1.2 Cosine Similarity of predicted word with options

In this approach, we use the article and question encoders as described in the first approach to encode the article and question. However, instead of using a binary classifier, we used a decoder layer to predict the missing word. We used the “@placeholder” token’s hidden embedding as an input to the decoder layer along with the context vector from question encoder and hidden weights from article encoder. This layer predicts a word from vocabulary which would fit in the place of “placeholder”. We then compute this word’s cosine-similarity with the given 5 options. The most similar option word is predicted as the answer. This method is similar

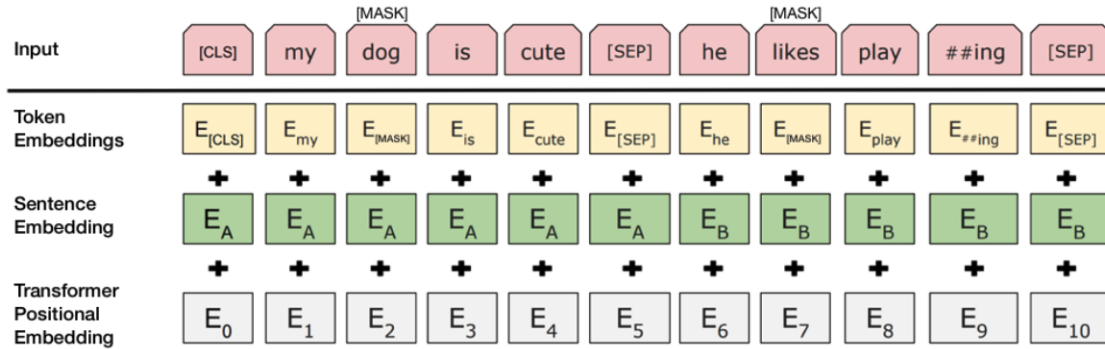


Figure 4: Masked Language Model (Image Src : (Devlin et al., 2018))

to an MLM based approach, and the first approach used above is somewhat similar to an MCQ based approach. This method gave slightly better results than the first approach, and thus, it gave us an idea that an MLM model should work better for our subtasks as compared to an MCQ based model.

## 4.2 Transformer based Models

### 4.2.1 Multiple choice Question Answering

The architecture of this approach is similar to that proposed by Radford (2018). We have a linear layer over a transformer model like BERT which takes the embedding of [CLS] token and calculates the cosine similarity of this token with given 5 options. Since, the [CLS] token represents the aggregate sequence representation (Devlin et al., 2018), it encodes the context of article and question together. The input sequence to the model is the concatenation of article and question (where the “@placeholder” is replaced by an option) delimited with the [SEP] token. On top of this, we have a softmax layer which calculates the score for each given option.

### 4.2.2 Masked Language Modeling

We used the transformer models like BERT and ALBERT for masked language modeling since they have been trained via the MLM objective. In this approach, the input sequence to our model is the concatenation of question and article tokens delimited with the [SEP] token where the “@placeholder” word in the question has been masked. We truncate the article from the end to fit into the maximum token sequence length. Since our task requires context reading from the article, we used different sentence embedding of the transformer models for question and article. An example of input sequence is given in Figure 4. The model’s output is a probability vector with probability scores of replacing

the masked token with any word in the vocabulary. We used the scores computed for the given 5 options and predicted option with the highest score as the correct answer.

We did an ensemble of BERT and ALBERT model predictions by taking the average score for each option predicted by these models. If the scores of BERT model predictions of the 5 options in a given example are  $B = \{B_1, B_2, B_3, B_4, B_5\}$  and the scores of ALBERT model predictions are  $A = \{A_1, A_2, A_3, A_4, A_5\}$ , then our Ensemble model gives the scores

$$S = \{S_i : S_i = \frac{A_i + B_i}{2} \forall i \in \{1, 2, 3, 4, 5\}\}$$

We later also did an ensemble of two ALBERT models where one is fine-tuned on the given subtask, and the other one is not. This gave us the best results on Subtask 1 and Subtask 2. However, we tried this approach in the post-evaluation phase and thus, we did not submit this system on the leaderboard.

Model	Task 1	Task 2
GAREader (baseline)	0.251	0.243
Binary Classification with Attention	0.201	0.212
Cosine Similarity approach	0.256	0.249

Table 2: Results of Encoder Based Approaches on Dev sets (Metric : Accuracy)

## 5 Experimental Setup

Our implementation uses the PyTorch library (Paszke et al., 2019) for deep learning models and the Transformers library by HuggingFace (Wolf et al., 2020) for the pre-trained transformer models and corresponding tokenizers.

	Model	Task 1	Task 2	Task 3 (Train 1, Val 2)	Task 3 (Train 2, Val 1)
1	BERT (MCQ approach)	0.195	0.202	0.201	0.198
2	BERT	0.6654	0.6523	0.475	0.449
3	BERT_Large - without Article	0.681	0.690	0.5347	0.5364
4	BERT_Large	0.7455	0.7403	0.6451	0.5913
5	ALBERT xxlarge - without Article - Not Finetuned	0.7011	0.7262	0.7262	0.7011
6	ALBERT xxlarge - Not Finetuned	0.8172	0.8190	<b>0.8190</b>	<b>0.8172</b>
7	ALBERT xxlarge - Finetuned	0.8291	0.8345	0.7426	0.6977
8	Ensemble (4 + 6)	0.8375	0.8401	0.7826	0.7729
9	Ensemble (6 + 7)*	<b>0.8685</b>	<b>0.8554</b>	0.8143	0.8064

Table 3: Transformer models’ results on Dev sets of each task (Metric : Accuracy)

\* Modified System after submission and not submitted in the task

We first experimented with the baseline model, Gated-Attention reader (Dhingra et al., 2016) provided by the task organizers. This model did not give good results, as shown in Table 2.

Then, we experimented with our Encoder based approaches as described in Section 4. We experimented with various loss functions like NLL loss, MSE and CrossEntropy loss. But, the results were poor for these methods too (Table 2). However, the Cosine similarity based approach (Section 4.1.2) performed slightly better than the Binary classification with Attention approach (Section 4.1.1) indicating that an MLM approach should work better than an MCQ approach.

To verify our claim, we experimented with the BERT Base model with the MCQ approach, which gave quite less accuracy, no better than a random prediction. However, the BERT model with the MLM approach performed way better on all the subtasks. We experimented with both large and small variants of BERT and ALBERT models where the large variants performed better as expected. We fine-tuned both the models without freezing any layers with Adam optimizer (Kingma and Ba, 2017). We fine-tuned the BERT model for 3 epochs and ALBERT model for 1 epoch. We used the learning rate of  $5e-5$  and a max-sequence length of 256 for both BERT and ALBERT. We also used pre-trained ALBERT model without any fine-tuning in some of our experiments.

We also experimented with the input sequence to understand and compare the degree of context-reading done in ALBERT and BERT models. We changed the input sequence to contain only ques-

tion tokens and then passed this sequence to our models. We then compared these results with the results obtained after passing the complete input sequence containing both article and question tokens. BERT gave an improvement of around 5-6% after passing the complete input sequence. However, ALBERT shows much more improvement of around 11-12% with the complete sequence. It shows that ALBERT’s training on a Sentence Ordering task is more effective for MLM tasks like ours than the BERT’s training on Next Sentence Prediction task.

We then experimented with the ensemble of BERT Large and ALBERT xxlarge-v2 model predictions. We experimented by assigning different weights to BERT and ALBERT models and found out that equal weights to both works better. We later did an ensemble of the fine-tuned ALBERT model with a non fine-tuned ALBERT model. It gave much improved results on Subtask 1 and Subtask 2.

## 6 Results and Analysis

The results of all the transformer based approaches are given in Table 3. The recurrence based models did not work and predicted answers with a random probability. We used GloVe (Pennington et al., 2014) vector embeddings that are not contextualized, unlike BERT embeddings. Moreover, the task requires some world knowledge since we need to predict an abstract word whose meaning can possibly be encoded if it is trained on large English corpus. Transformer based models are trained on large corpora and implicitly learn concepts grounded in

Type	Example	
WC	<b>Article</b>	... "Tennis chose me. It's something I never fell in love with," Tomic told Australia's Channel Seven. "Throughout my career I've given 100%. I've given also 30%. But if you balance it out, I think all my career's been around 50%." ....
	<b>Question</b>	Bernard Tomic says he has never "really tried" throughout his tennis career, adding that he has probably been @placeholder at "around 50%".
	<b>Options</b>	(A) held (B) aiming (C) honoured (D) operating (E) shown
	<b>Scores</b>	(A) 16.994 (B) 29.573 (C) 8.331 (D) 18.471 (E) 11.549
WN	<b>Article</b>	.... "These are home games that we have to win," McClaren said. "We are not performing individually and collectively the way that we did up until the Leicester replay. Has that taken too much out of us? I don't know. "We are not getting the rub of the green and we were doing that before. We are not scoring the first goal and we are not scoring goals....
	<b>Question</b>	Manager Steve McClaren says Derby County 's @placeholder have dropped and he has demanded an immediate response.
	<b>Options</b>	(A) chances (B) body (C) standards (D) side (E) artefacts
	<b>Scores</b>	(A) 28.372 (B) 7.169 (C) 27.527 (D) 10.246 (E) 8.395
CC	<b>Article</b>	Chiriac Inout was found in John Bright Street at about 23:30 GMT on 29 November, one of the coldest nights of the year. Police are investigating after CCTV appeared to show someone searching his pockets while he laid in a loading area behind The Victoria pub. An inquest date is yet to be fixed, the coroner's office confirmed.
	<b>Question</b>	A coroner has named a rough sleeper who may have had property @placeholder before he died in Birmingham city centre .
	<b>Options</b>	(A) lost (B) collapsed (C) stolen (D) delays (E) flowers
	<b>Scores</b>	(A) 13.214 (B) 12.342 (C) 27.909 (D) 2.336 (E) 4.510
CN	<b>Article</b>	.... The Blue Peter team say that Lindsey is safe and on her way back to dry land. Sport Relief said: "Lindsey's Sport Relief challenge was always going to be incredibly hard and zorbing many miles across the Irish Channel is a huge achievement. ....
	<b>Question</b>	Blue Peter 's Lindsey Russell has ended her attempt to cross the @placeholder between Northern Ireland and Scotland in a giant inflatable barrel for Sport Relief .
	<b>Options</b>	(A) gap (B) boundary (C) sea (D) title (E) difference
	<b>Scores</b>	(A) 24.295 (B) 26.728 (C) 26.874 (D) 4.482 (E) 18.486

Table 4: Some examples where our model makes mistakes or give correct results on Subtask 2 dev dataset. The options highlighted in blue are the correct answers and options highlighted in red are predicted by our model.

- \* WC - Wrong Confident - 27 such examples
- \* WN - Wrong Confused - 109 such examples
- \* CC - Correct Confident - 446 such examples
- \* CN - Correct Confused - 269 such examples

the world. Hence, transformer based approaches work better in our case.

The BERT model with the MCQ approach uses the embeddings of the '[CLS]' token to predict the correct option. But, it doesn't exploit the position of "@placeholder" token and hence it becomes difficult for the model to predict the correct result.

We observe that our Ensemble models work better on Subtask 1 and Subtask 2 as compared to the BERT and ALBERT models. However, on Subtask 3, the ALBERT model, which is not fine-tuned gives the best results. It owes to the fact that subtasks 1 and 2 differ a lot. If we fine-tune our model on one of the subtask, then it performs worse on the other.

For our final submission, we submitted our Ensemble model (8) (Table 3) for Subtask 1 and Subtask 2. We also submitted the fine-tuned ALBERT model on these subtasks. In Subtask 1, we are ranked 13th with our Ensemble model (8) with

an accuracy of 0.8212 on test set. In Subtask 2, we are ranked 11th with the fine-tuned ALBERT model with an accuracy of 0.8761 on test set. Surprisingly, in Subtask 2, fine-tuned ALBERT model performed better than our Ensemble model on the test set. This is possibly because our ensemble system performed only marginally better than fine-tuned ALBERT model on dev set. In Subtask 3, we submitted the non fine-tuned ALBERT model.

For understanding the mistakes made by our submitted Ensemble system, we analysed the confidence scores of our model's predictions. We performed the analysis of the system on the dev set of Subtask 2. We set a threshold factor (TF) of "1.4" for deciding between confident and confused predictions. If the confidence score of the model's predicted option is  $P$  and the score for the correct option word is  $T$ , then the model is confident in its



predicted answer if the following condition holds :

$$P \geq TF * T$$

It turns out that the model makes 50% confident predictions out of all the predictions in the dev set. Also, the model makes 20% wrong predictions confidently, while in 80% of the wrong predictions, model is confused between two options. In many cases, it is unable to understand the context properly and in a few cases, the model lacks the necessary world knowledge (Table 4). In first example in Table 4, the model predicts ‘aiming’ as the answer quite confidently. However, ‘operating’ is a more appropriate option due to the context given in the article. This example shows that our system fails to read the context properly in some cases. Also, consider the second example in Table 4. Here, the model is confused between two options : ‘chances’ and ‘standards’. Although, it is mentioned in the article clearly that Derby is performing poor in a past few matches, the model is not confident in predicting ‘standards’ which is the most suitable option here. It implies that our model is generally confused between all the option words that are semantically applicable in the question statement. Consider example 4 from Table 4. Here, the model is confused between option words ‘sea’ and ‘boundary’ because both of them fit well into the question. In order to make model more confident on such examples, we need to incorporate world knowledge into our system.

We also performed a similar post-competition analysis on our Ensemble System (9) (Table 3) and found out that it continues to make similar mistakes. But, in cases like example 1 in Table 4, it gives correct results. This is because, we used an ALBERT fine-tuned model instead of BERT fine-tuned model in this system which is better in context-reading as compared to BERT. Thus, this system gave slightly improved results.

## 7 Conclusion

The task of predicting abstract words with context from a question and article is quite novel in itself. We showed that this task can be modelled better as a masked language modeling task rather than multiple choice question answering task. The transformer based approaches worked best, where we used BERT and ALBERT models and their ensembles. These models are pretrained models and hence they perform better on our small dataset

after fine-tuning. We were able to improve the results of ALBERT model with our Ensemble model on subtasks 1 and 2, but on Subtask 3, the ALBERT model performs better. In future, we shall try to improve our results on Subtask 3. In our current approaches, we haven’t used options while training the model. We can try using a pairwise ranking loss function to rank the options according to their scores with a linear layer built on top of transformer models. This will help the model to predict answers more confidently and hence might also improve results. Moreover, we have used the same approach for Subtask 1 and 2. In future, we aim to incorporate common sense knowledge, for example, prototypical knowledge about activities in the form of scripts (Modi and Titov, 2014; Modi, 2016, 2017; Ostermann et al., 2018), or in the form of semantic networks like ConceptNet (Speer et al., 2018) for tackling two different definitions of abstractness and incorporating some knowledge in the two subtasks.

## References

- Mark Changizi, Andrew Hsieh, Romi Nijhawan, Ryota Kanai, and Shinsuke Shimojo. 2008. *Perceiving the Present and a Systematization of Illusions*. *Cognitive science*, 32:459–503.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. *CoRR*, abs/1810.04805.
- Bhuvan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. 2016. *Gated-Attention Readers for Text Comprehension*. *CoRR*, abs/1606.01549.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. *Teaching Machines to Read and Comprehend*.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. *The Goldilocks Principle: Reading Children’s Books with Explicit Memory Representations*.
- Diederik P. Kingma and Jimmy Ba. 2017. *Adam: A Method for Stochastic Optimization*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*.
- Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th*

- SIGLL Conference on Computational Natural Language Learning*, pages 75–83.
- Ashutosh Modi. 2017. Modeling common sense knowledge via scripts.
- Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 49–57.
- Tsendsuren Munkhdalai and Hong Yu. 2017. [Reasoning with Memory Augmented Neural Networks for Language Comprehension](#).
- Simon Ostermann, Ashutosh Modi, Michael Roth, Stefan Thater, and Manfred Pinkal. 2018. Mc-script: A novel dataset for assessing machine comprehension using script knowledge. *arXiv preprint arXiv:1803.05223*.
- Raghavendra Pappagari, Piotr Żelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. 2019. [Hierarchical Transformers for Long Document Classification](#).
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [PyTorch: An Imperative Style, High-Performance Deep Learning Library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.
- A. Radford. 2018. Improving language understanding by generative pre-training.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks](#).
- Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. [Conceptnet 5.5: An Open Multilingual Graph of General Knowledge](#).
- Otfried Spreen and Rudolph W. Schulz. 1966. Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention Is All You Need](#).
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [HuggingFace’s Transformers: State-of-the-art Natural Language Processing](#).
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

# ECNU ICA\_1 at SemEval-2021 Task 4: Knowledge-Enhanced Graph Attention Networks for Reading Comprehension of Abstract Meaning

Pingsheng Liu, Linlin Wang\*, Qian Zhao, Hao Chen, Yuxi Feng, Xin Lin\*, liang he

School of Computer Science and Technology,

East China Normal University, Shanghai 200062, China

{51205901014, 51205901129, 10175102236}@stu.ecnu.edu.cn

im0qianqian@ica.stc.sh.cn

{llwang, xlin, lhe}@cs.ecnu.edu.cn

## Abstract

This paper describes our system ECNU\_ICA\_1 for SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning. For this task, we utilize knowledge-enhanced Graph Attention Networks with a novel semantic space transformation strategy. It leverages heterogeneous knowledge to learn adequate evidences, and seeks for an effective semantic space of abstract concepts to better improve the ability of a machine in understanding abstract meanings of natural language. Experimental results show that our system achieves strong performance on this task in terms of both imperceptibility and nonspecificity.

## 1 Introduction

Recent years have witnessed the remarkable success of pre-trained language models in machine reading comprehension (MRC). Nevertheless, new research points out that these dominant approaches rely heavily on superficial text pattern-matching heuristics to achieve shallow comprehension on natural language (Zhang et al., 2020). For humans, the basic ability to represent abstract concepts guarantees an in-depth understanding of natural language. Consequently, teaching machines to better comprehend abstract meaning is a significant and urgent step to push the frontier technique of MRC forward.

If computers can understand passages as human do, we expect them to accurately predict abstract words that people can use in summaries of the given passages. Thus, researchers have recently proposed a reading comprehension of abstract meaning (ReCAM) task in SemEval 2021. Unlike some previous datasets such as CNN/Daily Mail (Hermann et al., 2015) that request computers to predict concrete concepts, e.g., named entities, ReCAM requires machines to fill out abstract words removed

from human written summaries. In ReCAM, subtask 1 and subtask 2 respectively evaluate the performance of machines towards imperceptibility and nonspecificity, two formal definitions of abstractness in natural language understanding (Spreeen and Schulz, 1966; Changizi, 2008). Specifically, concrete words refer to things, events, and properties that we can perceive directly with our senses (Spreeen and Schulz, 1966; Coltheart, 1981; Turney et al., 2011), e.g., donut, trees, and red. In contrast, abstract words refer to the ideas and concepts that are distant from immediate perception. Examples for abstract words include objective, culture, and economy. Subtask 1 requires machines to perform reading comprehension of abstract meaning for imperceptible concepts, while subtask 2 concentrates on hypernyms, which is more abstract and different from the concrete concepts (Changizi, 2008).

To better understand the abstract meaning, we utilize the Knowledge-Enhanced Graph Attention Network (KEGAT) architecture with a novel semantic space transformation strategy for ReCAM. It well incorporates structured knowledge base such as ConceptNet (Speer et al., 2017) and exploits a novel representation transformation strategy to improve the ability of machines in natural language understanding. The main contributions of our system are as follows:

- We utilize the KEGAT architecture to accomplish two subtasks in Reading Comprehension of Abstract Meaning, leveraging heterogeneous knowledge resources to provide adequate evidences and relying on Graph Attention Networks for the better reasoning.
- The proposed semantic space transformation strategy seeks for an effective representation mapping from concrete objects to abstract concepts, enabling machines to better understand the abstract meanings of natural language.

\*Equal corresponding authors.

- Extensive experiments show that our system achieves strong performance on this task in terms of both imperceptibility and nonspecificity.

## 2 Methodology

In this section, we describe the framework of our system and propose some strategies to enhance the reasoning ability of the model. An overview of the architecture is depicted in Figure 1.

### 2.1 Input Module

We cast the ReCAM task as a classification problem. For each instance, we assume that  $P$  is the passage,  $Q$  is the question,  $A$  is the number of candidate options, and  $O_i$  stands for the options, where  $i \in \{1, 2, \dots, A\}$ . For a specific training instance, we first replace the “@placeholder” in  $Q$  with  $O_i$ , and thus the resulting question-answer pair can be denoted as  $QO_i$ . Then we concatenate the passage and question-answer pairs as [CLS]  $P$  [SEP]  $QO_i$  [SEP], and denote this converted input as  $U_i$  for convenience. Although various approaches can be exploited to encode this  $U_i$ , we primarily adopt the basic way, in which tokens are represented with the one-hot vectors and the positional encoding is added, providing the model with a new embedding as  $E_{U_i}$  for every  $U_i$ .

### 2.2 Reasoning Module

Since pre-trained language models have achieved state-of-the-art performance in various NLP tasks (Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020), we adopt the pre-trained architecture to process the embedding  $E_{U_i}$  that is obtained from the previous step to get the high-level representation as  $\hat{E}_{U_i}^{base}$ . Specifically, we use Electra (Clark et al., 2020), a word-sensitive pre-trained language model which is composed of  $N$ -layer transformer encoders (Vaswani et al., 2017) depicted in the middle of Figure 1. Then, we utilize a Knowledge-Enhanced Graph Attention Network (KEGAT) component to accomplish the reasoning process based on all relevant entities and the high-level representation of the entire question-answer pair from the pre-trained model. The working principle of our KEGAT model is introduced later.

As shown in Figure 1, our KEGAT model mainly consists of a Graph Attention Network, a self-attention submodule and a multi-layer perceptron (MLP). It enables a multi-level reasoning process

from entities to sentences. For the entity level, we utilize some structured knowledge from ConceptNet with a different integration approach to achieve the goal of conducting inferences over new constructed subgraphs. Here, we adopt the N-gram method to extract all entities from the converted input  $U_i$ , and use edge weight as the probability to select a maximum of  $k$  adjacent nodes from ConceptNet for subgraph construction. Suppose the number of entities is  $n$ , we construct  $n$  subgraphs in total, and the subgraphs may be connected with edges. Next, we utilize the conceptnet-numberbatch\* to obtain the  $i$ -th entity embedding as the initial representation  $h_i^{(0)}$ , which is subsequently refined by the  $L$ -layer Graph Attention Network (GAT). In the refinement process, the GAT module automatically learns an optimal edge weight between two entities in these subgraphs based on the ReCAM task, indicating the relevance of adjacent entities to every central entity. In other word, for a central entity, the GAT tries to only assign higher weight values to those edges connected with several most reasonable adjacent entities from the constructed subgraph, and discards some irrelevant edges. Thus, the abstract semantic inference ability of our model is highly improved with the knowledge incorporated by the refined subgraphs. The working principle of our GAT is in Eq. 1–3.

$$h_i^{(l+1)} = \sigma \left( \frac{1}{M} \sum_{m=1}^M \sum_{j \in \mathcal{N}_i} \alpha_{ij_m}^{(l)} \mathbf{W}_m^{(l)} h_j^{(l)} \right) \quad (1)$$

$$\alpha_{ij}^{(l)} = \text{softmax}_j \left( f([\mathbf{W}^{(l)} h_i^{(l)}; \mathbf{W}^{(l)} h_j^{(l)}]) \right) \quad (2)$$

We update each entity node based on Eq. 1, where  $\sigma(\cdot)$  represents a ELU function (Clevert et al., 2016),  $\mathbf{W}$  is the network parameter,  $h_i^{(l)}$  is the representation from the  $l$ -th layer of GAT, and  $\mathcal{N}_i$  stands for all adjacent nodes to the  $i$ -th entity.  $M$  is the number of independent attention mechanisms in Eq. 2, and  $\alpha_{ij}^{(l)}$  is the relevance degree of the  $j$ -th adjacent entity with respect to the  $i$ -th entity. Besides,  $f(\cdot)$  represents a projection function converting the vector to a real number, and  $[\cdot]$  stands for the concatenation operation. Finally, we define

$$\hat{E}_{U_i}^{gnn} = \frac{1}{n} \sum_{i=1}^n h_i^{(L)} \quad (3)$$

to be the final representation for entity subgraphs that are obtained from the GAT.

\*ConceptNet-Numberbatch:  
<https://github.com/commonsense/conceptnet-numberbatch>

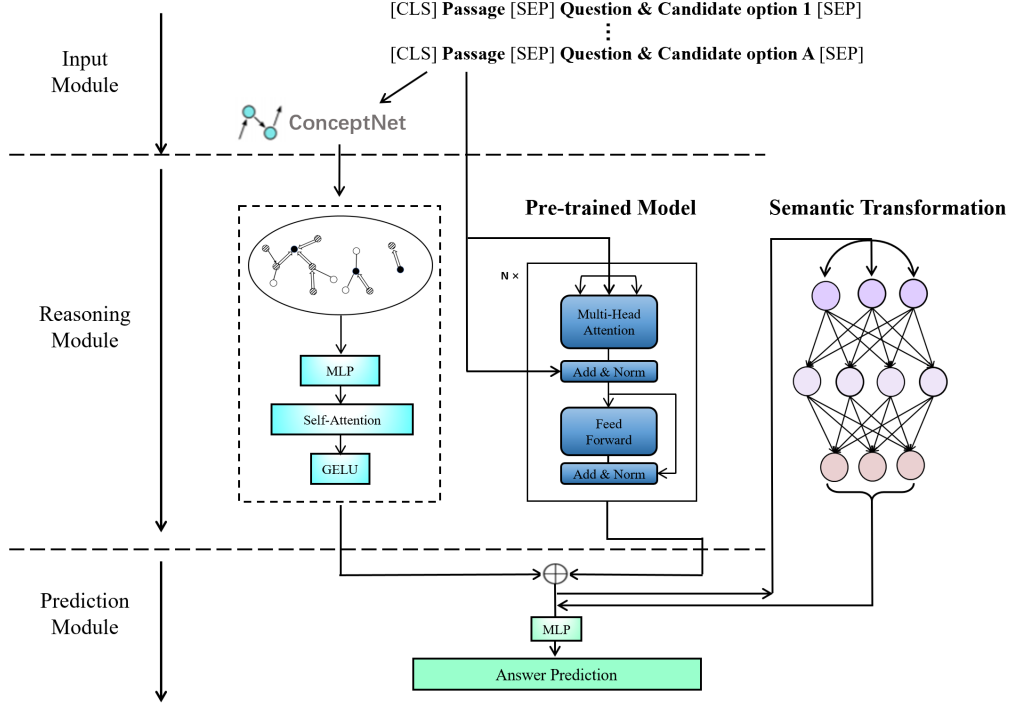


Figure 1: The overview of our system for ReCAM.

From the sentence level, we adopt a self-attention submodule and several MLPs to promote the model to reason over both entities and input sentences. We first utilize a MLP to fuse the symbolic and semantic representations and then take a self-attention operation for refinement. Thus, the entity-level representation can be further refined by taking the question-answer pair as a reference. To sum up, some valuable dimensions can be highlighted to retain the most reasonable information from the fused representations  $\hat{E}_{U_i}^{all}$  to improve the reasoning ability. We formulate these steps as Eq. 4 and Eq. 5.

$$\hat{E}_{U_i}^{all} = \text{MLP}([\hat{E}_{U_i}^{base}; \hat{E}_{U_i}^{gnn}]) \quad (4)$$

$$G_{U_i} = \sigma(\text{SelfAttn}(\hat{E}_{U_i}^{all})) \quad (5)$$

where  $G_{U_i}$  is the refined representation,  $\text{SelfAttn}(\cdot)$  represents a self-attention operation, and  $\sigma(\cdot)$  is the activation function. Finally, we concatenate  $G_{U_i}$  and  $\hat{E}_{U_i}^{base}$  to obtain the entire reasoning representation as

$$\hat{E}_{U_i} = [G_{U_i}; \hat{E}_{U_i}^{base}] \quad (6)$$

### 2.3 Prediction Module

With the previous multi-level reasoning process, we obtain the representation of converted inputs as  $\{\hat{E}_{U_i}\}_{i=1}^A$  for each instance. In the prediction module, we use a multi-layer perceptron to solve

the downstream tasks of ReCAM based on Eq. 7–9.

$$P_i = \text{MLP}(\hat{E}_{U_i}), \quad P' = \text{softmax}(P) \quad (7)$$

$$y = \arg \max(P') \quad (8)$$

$$\mathcal{L} = - \sum_{i=1}^A y_i^* \log P'_i \quad (9)$$

where  $y$  represents the prediction result, and  $P'_i$  stands for the probability of selecting the  $i$ -th option label.  $P$  is the output of the MLP, where  $P \in \mathbb{R}^{A \times 1}$ .  $\mathcal{L}$  is the training objective to minimize negative log-likelihood and  $y^*$  here stands for one-hot vector of the optimal label.

### 2.4 Adaptive Strategies

**Noise Reduction Strategy** Previous methods of knowledge integration often lead to inevitable noise (Zhong et al., 2019; Wang et al., 2019), and it is still an open research problem to balance the impact between noise and the amount of incorporated knowledge. (Weissenborn et al., 2018; Khashabi et al., 2017). Our KEGAT can alleviate the noise that is caused by incorporated structured knowledge to a certain extent. This module accomplishes the goal of identifying the most reasonable external entities and discarding the irreverent ones. For

example, we rely on both entity-level and sentence-level inference thoroughly that is discussed in the previous Reasoning Module part to achieve this goal. Furthermore, we remove several unimportant types of edges to avoid unnecessary noises, such as `"/r/DistinctFrom"`, `"/r/ExternalURL"`, etc.

**Semantic Space Transformation Strategy** Unlike some previous MRC tasks that request computers to predict concrete concepts, ReCAM task here asks models to fill out abstract words removed from human written summaries. Thus, we utilize a semantic space transformation strategy to convert ordinary semantic representation into abstract representation for classification. Specifically, for the final answer prediction, this approach deals with the hidden vector representation  $V$  which is obtained ahead of the prediction module. One method is to extend the dimension (ED) of  $V$ . For instance, we use a MLP to expand  $V$  by 500 dimensions and then perform the downstream classification prediction. The second attempt is to transform  $V$  directly with a nonlinear activation function, such as RELU. And another method is to transform  $V$  through a simple deep neural network (DNN), which is depicted in the right of Figure 1.

### 3 Experiments

#### 3.1 Datasets and Metric

In the ReCAM task, it requires the model to fill out abstract words removed from human written summaries. The total number of abstract words that can be selected is five. We utilize Accuracy as a metric to evaluate model performance.

#### 3.2 Experimental Settings

In our experiment, we set the maximum sentence length as 210 and the batch size as 16. During training, we freeze all layers and learn 2 epochs with a learning rate of 0.001 except for the last classification layer. In the fine-tuning phase, we unfreeze all layers and learn 10 epochs with a learning rate of 0.000005. Like the training phase, it is beneficial to use the weights of the pre-trained language model to correct the randomly initialized classification layer. All layers of the entire model in the fine-tuning phase are suitable for classifying downstream tasks with the low learning rate. For each phase, we save model parameters when it reaches the highest accuracy on the dev set, and load it at the beginning of the next phase. In addition, we

adopt the Adam optimizer (Kingma and Ba, 2015) and set epsilon to be 0.000001 for the gradient descent. We train our model with Titan XP GPUs.

#### 3.3 Results

Table 1 shows the results of the top five teams from the leaderboard for ReCAM task (by February 10). Our system achieves the 3rd place in Subtask 1 in terms of Accuracy. And it can be concluded from Table 2 that our system has the ability to solve the ReCAM task.

Besides, we test the performance of our system with the strategies mentioned in Section 2.4. Here, `+KEGAT` represents our proposed model with Knowledge-Enhanced Graph Attention Networks, `+ED`, `+RELU`, `+DNN` refer to our system with different semantic space transformation strategies. In addition, Dev Acc. and Test Acc. stand for the accuracy on the dev set and test set respectively. Table 2 shows the experimental results of our system on the ReCAM task. In this table, the baseline model GA Reader provided by the competition organizer is not ideal, and its performance is slightly higher than 20% with our actual testing. We conclude that on the dev set, our system respectively achieves the relative improvement of 6.69% and 4.24% on subtask 1 and subtask 2 when adding KEGAT submodule compared with the fine-tuned Roberta large. Moreover, we test the performance of three ensemble models shown in the bottom of 2, and the `Electra-large ED + Electra-large KEGAT-RELU` ensemble obtains the best performance on the dev set, which respectively outperforms the fine-tuned Roberta large model with the relative improvement of 7.41% and 5.29% on subtask 1 and subtask 2. Here, this ensemble framework refers to the combination of two models. Therefore, it can be concluded that the ensemble models with the semantic space transformation strategy greatly improve the reasoning ability of our system, and the single system with multiple strategies performs well in most cases.

#### 3.4 Further Discussion

To further investigate this task, we have additionally assessed the impact of data bias on the model performance. By statistics, the average length of passages in the dev sets of subtask 1 and subtask 2 are 268.8 and 434.6, respectively. In general, longer passages often consist of more noise that greatly influences answer reasoning process of the model. We only select a portion of contents from

Subtask 1			Subtask 2		
Rank	Team Name	Accuracy	Rank	Team Name	Accuracy
1	Silvilla	95.11	1	PINGAN Omini-Sinitic	95.29
2	PINGAN Omini-Sinitic	93.04	2	Silvilla	94.89
<b>3</b>	<b>ECNU_ICA.1 (ours)</b>	<b>90.47</b>	3	tt123	93.41
4	tt123	89.98	<b>4</b>	<b>ECNU_ICA.1 (ours)</b>	<b>93.01</b>
5	cxn	88.69	5	cxn	92.91

Table 1: Top 5 results for ReCAM task.

Model	Subtask 1		Subtask 2	
	Dev Acc.(%)	Test Acc.(%)	Dev Acc.(%)	Test Acc.(%)
GA Reader	24.61	-	22.79	-
<i>Our Architectures</i>				
-w fine-tuned Roberta-large	85.18	-	87.30	-
-w Electra-large	90.80	89.28	91.07	90.48
-w Electra-large + KEGAT	91.87	89.37	91.54	92.01
-w Electra-large + KEGAT-RELU	92.35	90.37	91.89	92.11
-w Electra-large + ED	91.51	90.12	91.65	90.95
-w Electra-large + DNN	91.40	-	91.77	-
<i>Ensemble Models -w Electra-large</i>				
+ KEGAT	91.99	-	92.36	-
ED + KEGAT	92.47	-	92.48	-
ED + KEGAT-RELU	<b>92.59</b>	<b>90.47</b>	<b>92.59</b>	<b>93.01</b>

Table 2: Experimental results of ReCAM task.

	Subtask1	Subtask2
<b>AVG length</b>	268.8	434.6
<b>Position</b>	<b>Dev Acc.(%)</b>	<b>Dev Acc.(%)</b>
0-210	90.80	91.07
211-420	89.31	89.65

Table 3: Performance on different contents of passage.

the given passage for this assessment instead of the whole passage. Specially, in the given dataset, we take a fixed length of 210 as the content interval by intercepting it at two different positions, namely token ID 0 ~ 210 and token ID 211 ~ 420. Then we fine-tune the Electra-large model for each subtask using their own training set and compare the performance of the fine-tuned Electra model on two different passage intervals. It means that we have conducted experiments with different passage contents twice. Table 3 reports the results of our system on these different passage intervals. In this

table, compared to the experiment that adopts the passage content with position from 0 to 210, intercepting the one with position from 211 to 420 leads the performance to drop by about 1 ~ 2% on these two subtasks. Thus, we conclude that the positional bias indeed affects model performance to some extent.

## 4 Conclusion

We utilize a knowledge-Enhanced Graph Attention Network architecture with semantic transformation strategies for machines to better comprehend the abstract meanings of natural language. It well incorporates heterogeneous knowledge and relies on Graph Attention Networks to learn adequate evidences. The subsequent semantic transformation enables an effective representation mapping from concrete objects to abstract concepts. Our system achieves strong performance on this comprehension task in terms of both imperceptibility and non-specificity. We hope this work can shed some lights on the study of in-depth reading comprehension.

## Acknowledgements

This work was supported by the National Innovation 2030 Major ST Project of China, the Fundamental Research Funds for the Central Universities, National Natural Science Foundation of China (No.62006077), Shanghai Sailing Program (No.20YF1411800), and the Science and Technology Commission of Shanghai Municipality (No.20511105102).

## References

- Mark A. Changizi. 2008. [Economically organized hierarchies in wordnet and the oxford english dictionary](#). *Cognitive Systems Research*, 9(3):214–228.
- Kevin Clark, Minh-Thang Luong, V. Quoc Le, and D. Christopher Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *ICLR*.
- Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2016. [Fast and accurate deep network learning by exponential linear units \(elus\)](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Max Coltheart. 1981. [The mrc psycholinguistic database](#). *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS’15*, page 1693–1701, Cambridge, MA, USA. MIT Press.
- Daniel Khoshdel, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2017. [Learning what is essential in questions](#). In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 80–89, Vancouver, Canada. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [Albert: A lite bert for self-supervised learning of language representations](#). In *International Conference on Learning Representations*.
- Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI’17*, page 4444–4451. AAAI Press.
- Otfried Spreen and Rudolph W. Schulz. 1966. [Parameters of abstraction, meaningfulness, and pronouncability for 329 nouns](#). *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.
- Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. [Literal and metaphorical sense identification through concrete and abstract context](#). In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 680–690, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Xiaoyan Wang, Pavan Kapanipathi, Ryan Musa, Mo Yu, Kartik Talamadupula, Ibrahim Abdelaziz, Maria Chang, Achille Fokoue, Bassem Makni, Nicholas Mattei, and Michael Witbrock. 2019. [Improving natural language inference using external knowledge in the science questions domain](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7208–7215.
- Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2018. [Dynamic integration of background knowledge in neural NLU systems](#).
- Zhilin Yang, Zihang Dai, Yiming Yang, G. Jaime Carbonell, Ruslan Salakhutdinov, and V. Quoc Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 32 (NIPS 2019)*, pages 5754–5764.
- Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020. Machine reading comprehension: The role of contextualized language models and beyond. *arXiv preprint arXiv:2005.06249*.
- WanJun Zhong, Duyu Tang, Nan Duan, Ming Zhou, Jiahai Wang, and Jian Yin. 2019. Improving question answering by commonsense-based pre-training. In *Natural Language Processing and Chinese Computing*, pages 16–28, Cham. Springer International Publishing.



# LRG at SemEval-2021 Task 4: Improving Reading Comprehension with Abstract Words using Augmentation, Linguistic Features and Voting

Abheesht Sharma\*

Dept. of CS&IS  
BITS Pilani, Goa Campus

f20171014@goa.bits-pilani.ac.in

Harshit Pandey\*

Dept. of Computer Science  
Pune University

hp2pandey1@gmail.com

Gunjan Chhablani\*

Dept. of CS&IS  
BITS Pilani, Goa Campus

chhablani.gunjan@gmail.com

Yash Bhartia

Dept. of CS&IS  
BITS Pilani, Goa Campus

f20190151@goa.bits-pilani.ac.in

Tirtharaj Dash

APP Centre for AI Research  
BITS Pilani, Goa Campus

tirtharaj@goa.bits-pilani.ac.in

## Abstract

In this article, we present our methodologies for SemEval-2021 Task-4: Reading Comprehension of Abstract Meaning. Given a fill-in-the-blank-type question and a corresponding context, the task is to predict the most suitable word from a list of 5 options. There are three sub-tasks within this task: Imperceptibility (subtask-I), Non-Specificity (subtask-II), and Intersection (subtask-III). We use encoders of transformers-based models pre-trained on the masked language modelling (MLM) task to build our Fill-in-the-blank (FitB) models. Moreover, to model imperceptibility, we define certain linguistic features, and to model non-specificity, we leverage information from hypernyms and hyponyms provided by a lexical database. Specifically, for non-specificity, we try out augmentation techniques, and other statistical techniques. We also propose variants, namely *Chunk Voting* and *Max Context*, to take care of input length restrictions for BERT, etc. Additionally, we perform a thorough ablation study, and use Integrated Gradients to explain our predictions on a few samples. Our best submissions achieve accuracies of 75.31% and 77.84%, on the test sets for subtask-I and subtask-II, respectively. For subtask-III, we achieve accuracies of 65.64% and 62.27%. The code is available [here](#).

## 1 Introduction

A very common assessment in schools is question-answering based on a given “comprehension passage”. Students are given a comprehension passage, from which they are supposed to glean necessary information, and answer short questions (such as fill-in-the-blanks-type question) based on what they have garnered from the given passage. While trying to find the most appropriate word for the blank, the children look at the words surrounding the blank

(“context”). The word should be such that when the word fills the blank, the sentence makes sense and it is grammatically correct. Inspired by this, and perhaps, after the enormous success of Transformers (Vaswani et al., 2017), researchers at Google came up with a large number of “pretraining tasks” and built knowledge-heavy language models which could be fine-tuned on various natural language processing (NLP) downstream tasks. One of the earlier pretraining tasks was “Masked Language Modelling (MLM)”, one of the two pretraining tasks of the breakthrough model, BERT (Devlin et al., 2019). The approach here was similar to how kids are taught language at school: some tokens in the text were randomly “masked” and the model was trained to predict these masked tokens.

SemEval-2021 Task-4 (Zheng et al., 2021) focuses on a similar idea. Every sample has an article, and a corresponding question. The question has a blank which the model is supposed to predict from a set of 5 options. The novelty in the task lies in its 3 subtasks: Imperceptibility (subtask-I), Non-Specificity (subtask-II), and Intersection (subtask-III). A description of these subtasks is given in Section 3. In this work, we propose using BERT and its derivative models such as DistilBERT (Sanh et al., 2019), ALBERT (Lan et al., 2019) and RoBERTa (Liu et al., 2019). Further, we propose 2 BERT variants: (1) *BERT Voting*; (2) *BERT Max. Context*. Most importantly, we also model the concepts of imperceptibility and non-specificity. For imperceptibility, we create statistical embeddings using features that have a high correlation with concreteness. For non-specificity, we propose two approaches: (1) we augment the dataset by replacing some nouns in the article by their hypernyms; and (2) we use the options’ hyponyms to decide the most appropriate option. We also experiment with GA-Reader (Dhingra et al., 2017b) and GSAMN-based approaches (Lai et al., 2019) by trying out

\* Equal contribution. Author ordering determined by coin flip.

their various combinations with BERT.

In Section 2, we perform a succinct literature survey. Section 3 elucidates our approach, including the modelling aspect, the various variants of the base model, and the different ways we model imperceptibility and non-specificity. In Section 4, we perform an extensive ablation and comparative study.

## 2 Background

The advent of large-scale question answering systems began with straightforward tasks, like the one introduced by the SimpleQuestions Dataset (Bordes et al., 2015), which consisted of knowledge-base fact triples which were later used to answer questions. However, this dataset would only judge a model based on the ability to relate the facts to the question at hand. The purpose of NLP research is to be able to create a generalised model that may answer questions based on any context, thus datasets like the CNN Daily Mail (Hermann et al., 2015) and SQuAD (Rajpurkar et al., 2016) were created. In a typical question-answering dataset, an original and anonymised context is provided before each question. Before transformers, methods consisting of LSTM/GRUs were used to achieve good results on the aforementioned tasks. These datasets however, always had answers in the passage.

The CLOTH (Xie et al., 2018) dataset focuses on passages from middle-school and high-school text, with multiple fill-in-the-blanks in the passage. The ReCAM (Zheng et al., 2021) dataset puts a twist to archetypal fill-in-the-blank datasets by providing answer choices that are abstract in some form and which are not available in the passage itself. The models created for the QA task have to take into account semantic relations between the options and the context. GA-Reader (Dhingra et al., 2017b), is one such model, which utilises a multi-hop architecture with a novel attention mechanism, that serves as a baseline to this task.

## 3 Methodology

### 3.1 MLM-Based Transformers for Cloze-Style QA

The first model we employ follows a cloze-style question answering approach, in which we use various pretrained transformer models as encoders, followed by a decoder layer, which helps us to select the correct answer.

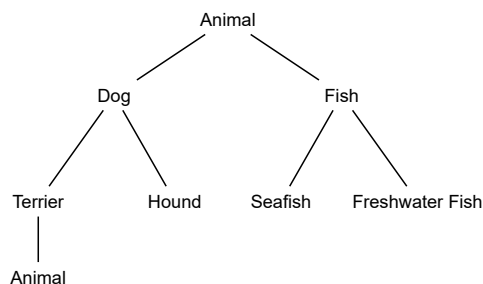


Figure 1: An example of a Hypernymy Tree

Specifically, we leverage BERT along with some of its popular and successful variants such as: DistilBERT, ALBERT, and RoBERTa. In the MLM task, tokens in the text are randomly masked, and the model is trained in a self-supervised way to predict these masked tokens. Conceptually, these transformers-based models are expected to take care of bidirectional context while predicting the masked token.

In our method, firstly, the transformer model learn the contextual embeddings of the article and the question. For the next block, the embedding of the masked token (i.e., the blank) is passed through a fully-connected layer, of which, the number of outputs corresponds to the size of the vocabulary space for the pretrained model. Each candidate option is first tokenised using WordPiece tokeniser (Wu et al., 2016), and mapped to the vector in the output vocabulary space. If the candidate option generates multiple tokens, we average the mapped scores. The model chooses the option with the highest logit value. An overview of the model is given in Figure 2.

### 3.2 Improvement Approaches

#### 3.2.1 Imperceptibility:

Nouns can be clearly demarcated into two broad categories: Concrete Nouns, and Abstract Nouns. Concrete Nouns are words that represent tangible concepts, i.e., any noun referring to a name, place, object, material, etc. is considered a concrete word. Concrete words refer to concepts that can be felt by 5 human senses: Sight, Sound, Smell, Taste, and Touch. In contrast, any noun alluding to an abstract concept that cannot be experienced by our senses is an abstract word (Spreeen and Schulz, 1966). In subtask-I, the model has to predict the most accurate and the most imperceptible word from the given options. To model the imperceptibility of

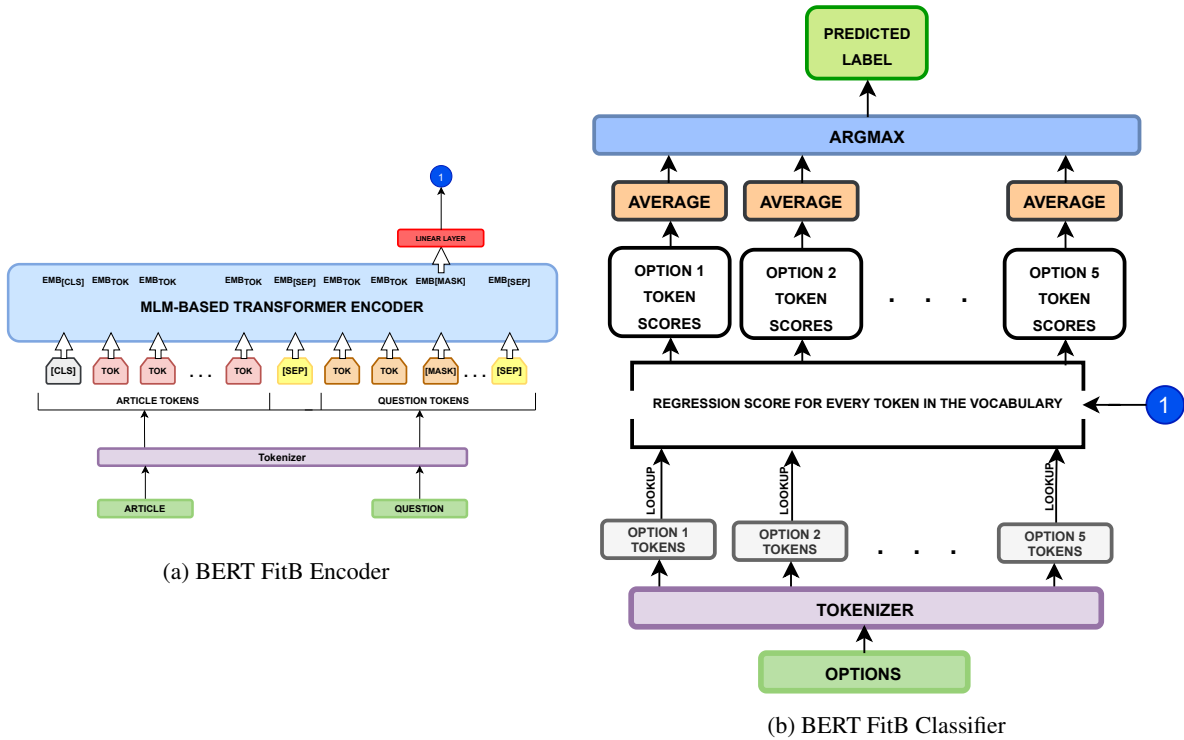


Figure 2: Architecture of Transformer-based FitB Model

every word, we incorporate certain linguistic features which are highly correlated with the notion of “imperceptibility”. These linguistic features are listed below:

**Length and Frequency of the Word** In existing literature, authors have claimed that there exists strong evidence that concrete words are, in general, shorter than abstract words (Tanaka et al., 2013). A reasonable justification provided is that more frequently used words tend to be short (Feng et al., 2011) and since humans have a penchant for describing objects, places, or things near them, these frequently used words are generally concrete nouns. It is rather intuitive that humans would prefer ease in the pronunciation of oft-used words. Moreover, many abstract words in the English language are formed by adding suffixes to the root word, such as “coarse” becomes “coarseness”, “forget” becomes “forgetfulness” and so on (Tanaka et al., 2013).

**Number of Senses of the Word** In Linguistics, polysemy refers to the capacity for a word to have multiple meanings or senses. Abstract nouns are observed to be more “polysemous” than concrete nouns (Tanaka et al., 2013). For example, in WordNet (Fellbaum, 1998), the word “dog” has 8 senses, while the word “love” has 10 senses.

**Number of Hyponyms** Tanaka et al. 2013 find a direct correlation between the abstractness of a noun and the number of hyponyms the word has. We consider the number of hyponyms of the most commonly occurring sense of the word, and the average number of hyponyms of all the senses of the word.

**Score-based Features** Abstract nouns evoke emotions in humans. SentiWordNet (Baccianella et al., 2010), another lexical database like WordNet, gives scores based on the how positive, negative or objective they are. Abstract words have a higher positive/negative score, while concrete words have a higher objective score. Again, here, we consider these scores for the most commonly occurring sense, and the average scores of all the senses of the word.

**Depth in Hypernymy Tree** This feature is more suited for non-specificity. However, we include this as a feature of imperceptibility since the concepts of imperceptibility and non-specificity are related. For example, consider the words “money” and “property”. The latter is more imperceptible and non-specific than the former. Moreover, this is particularly useful for Subtask-III. Therefore, the depth of a word in the hypernymy tree is directly proportional to the concreteness of the word.

From the features above, we have a 13-dimensional vector for every word in the lexicon. The embedding is created so that every dimension is directly proportional to the concreteness of the word. For example, the length of a word is in general, indirectly proportional to the concreteness of the word, so we take the length dimension of the vector as *large value – length of word*, where we take 10,000 as the *large value*. The large value chosen was the same for all features which are indirectly proportional to concreteness.

Towards improving the trained model, we use a method which we term as the *Difference Method*. If the difference of the top-2 probabilities predicted by the model is greater than a certain threshold, this implies that the model is sure of the prediction it has made. However, if the difference is less than the tunable threshold, the model is ambivalent about whether the option with the highest probability or the option with the second highest probability is correct. In this case, we compute for how many dimensions the value of the linguistic embedding of the second word is less than the value of the linguistic embedding of the first word. If the majority of the values (i.e., 7) are less, we change the prediction of the model to the second-most probable option. The threshold is tuned on the dev set. Furthermore, we use a *Threshold Method* towards improving the model performance. If the highest probability is less than a tunable threshold, the model is unsure of its predictions and we consider the improvement approaches on the option with the second-highest probability.

### 3.2.2 Non-Specificity

According to [Spreen and Schulz, 1966](#), a highly specific word refers to a very particular instance, while a non-specific word refers to a generic concept, i.e., it encompasses many classes/instances. For example, consider the words “animal”, “bird” and “eagle”. The words are listed in increasing order of specificity.

We find parallels between the definition of specificity/non-specificity and the linguistic phenomenon of hypernymy. [Schreuder and Baayen, 1995](#) define a hypernym as “a word with a general meaning that has basically the same meaning of a more specific word”. The more specific word is the corresponding hyponym. In simpler terms, each word is related to some super-types and sub-types, called as hypernyms and hyponyms, respectively. In linguistics, hyponymy is a semantic relation be-

tween a hyponym denoting a subtype and a hypernym denoting a supertype.

For example, in figure 1, as we traverse up the hypernymy tree, assuming we consider the word “dog”, we find that its hypernym is “animal”, which is much broader than “dog”. On the other hand, as we go down the hypernymy tree, we find more specific terms for the word “dog” such as “terrier”. Essentially, hyponyms represent “IS-A” relationships. For example, “terrier” is a “dog”. We leverage the hypernymy property of words to help the model in deciding the most non-specific option. The two methods which we implement are:

**Hypernym Augmentation Method** In order to infuse a sense of non-specificity (other than training on the given dataset for non-specificity), we augment the dataset for subtask-I. We randomly select  $n$  nouns from the article by using a basic POS Tagging pipeline. For each noun, we use the Lesk algorithm ([Lesk, 1986](#)) to find the most appropriate sense of the word based on its context. For this sense of the word, we find its hypernyms, pick a hypernym uniformly at random from this list of hypernyms and replace the noun in the article with the hypernym. We do this for all  $2^n$  combinations, i.e., corresponding to every sample, we have  $2^n$  augmented samples. Furthermore, we randomly mask tokens in this dataset and train BERT on the MLM task, on this dataset. This serves a dual purpose. Firstly, it serves as a sort of domain adaptation, and secondly, it infuses a sense of non-specificity in the model.

While finetuning BERT MLM on the augmented dataset, we freeze two layers, due to time and computational constraints. We replace the normal BERT Encoder in our BERT FitB model with the BERT Encoder fine-tuned on the augmented dataset.

**Hyponyms Options Method** Here, we use the *Difference Method/Threshold Method*. If the model is sure of its prediction, we keep the prediction of the model. Otherwise, we generate hyponyms for each option using WordNet. After the hyponyms are tokenised, we use the trained model’s output and map each hyponym token to the output vocabulary space and get the corresponding scores. We then take the maximum score amongst all of the hyponyms as the predicted probability for that option. The reason for incorporating this approach pertains to how the transformer models were pre-

trained. Consider the following sentence: “He had a [MASK] and it was bitter”. Now, suppose that we have two options: “beer” and “drink”. Generally, our transformer-based model would look at the word “bitter” and predict “beer”. However, “drink” is more non-specific than “beer”.

### 3.2.3 BERT Fill-in-the-blanks Variants

To address the limitations of the vanilla transformer-based models, we attempt multiple modifications to the proposed baseline transformer models, specifically for BERT. The major limitation of the pretrained BERT model that we’ve used, is the restriction on the length of the tokenised inputs. Only 512 tokens from a sample can be processed by BERT in one parse and hence, some articles end up getting truncated and context is lost. The following are some of the modifications we’ve made to improve the performance of our models:

**Voting** We tokenise the question and the article. We split the article into chunks and pair each chunk with the question such that the length of the tokenised  $(chunk, question)$  pair is 512. While splitting the article into chunks, we keep a max-overlap stride of 128 so that the context of the previous chunk is not lost. We train the model on these newly formed  $(chunk, question)$  pairs. During inference, we take the weighted sum of the logits. For *BERT FitB Voting (Similarity)*, the weights are calculated as:

$$weight_{ij} = \frac{u_i \cdot v_j}{\|u_i\| \|v_j\|} \quad (1)$$

where  $u_i$  is the embedding of the question in the  $i^{th}$  sample, and  $v_j$  is the embedding of the  $j^{th}$  chunk of the sample’s article. To find the embeddings, we extract the [CLS] embedding from a pretrained BERT encoder.

We also try out an alternate way of defining the weights:

$$weight_{ij} = \frac{|\{q_i \text{ toks.}\} \cap \{chunk_j \text{ toks.}\}|}{|\{chunk_j \text{ toks.}\}|} \quad (2)$$

where  $\{q_i \text{ toks.}\}$  is the set of tokens in the  $i^{th}$  sample’s question, and  $\{chunk_j \text{ toks.}\}$  is the set of tokens in the  $j^{th}$  chunk of the sample.  $|\cdot|$  represents the cardinality of a set. We call the method *BERT FitB Voting (Exact Matching)*.

We normalise the computed weights:

$$norm\_weight_{ij} = \frac{weight_{ij}}{\sum_{j=1}^{n_i} weight_{ij}} \quad (3)$$

where  $n_i$  is the number of chunks in the  $i^{th}$  sample.

The idea behind this is that higher the similarity between the question and the article’s chunk, higher is the weight assigned to the logits returned by the trained model with the question-chunk pair as input. In Equation 2, we find the fraction of tokens common between the question and chunk.

**Max Context** This method is a slight modification of the Voting Method. Instead of training the model on all  $(chunk, question)$  pairs for a particular sample, we train the model on the pair with the highest weight. The weights are calculated as described in Equation 2.

### 3.2.4 GA-Reader-based Approaches

We propose a few modifications to the baseline, namely GA-Reader (Dhingra et al., 2017a) provided by the organisers.

**GA-Reader BERT** We use GA-Reader on top of BERT embeddings. This could lead to potential improvement in performance for subtask-I as BERT embeddings are more feature-rich than GloVe embeddings.

**GA-BERT** Based on the Gated-Attention Reader, we came up with an approach that uses Gated-Attention across two-BERT streams. The first stream takes in the question input, and works like the regular BERT model. The second stream takes the article input. Assume the layer outputs for layer  $L$  are  $Q_L$  and  $A_L$ , respectively, for question and article streams. Then, to the layer  $L + 1$  for question stream,  $Q_L$  is passed as input, while to layer  $L + 1$  for article stream,  $GA(Q_L, A_L)$  is passed, where  $GA$  is the Gated-Attention function. This is done for all 12 layers of BERT-BASE. Finally, on this model, two types of heads are attached - Selection and Pooling (similar to BERT FitB), and Attention Classification (similar to GA-Reader). The logits for each head are concatenated and a fully-connected layer is added on top. Since this is a major change in the architecture of BERT, this model needs a significant amount of pretraining.

**Answer-Attention** Since GA-Reader also attends to the candidate answer embeddings, we also attempt an approach where we pass the options to the BERT model. On the option embeddings and the [MASK] token embeddings, we apply multiplicative attention (dot product) to get attention

Model Information		Imperceptibility		Non-Specificity	
Model	Variant	Val Acc.	Test Acc.	Val Acc.	Test Acc.
BERT Fill-in-the-Blank	base	67.03%	66.77%	64.39%	65.74%
BERT Fill-in-the-Blank	large	<b>74.79%</b>	<b>75.30%</b>	<b>72.73%</b>	<b>75.16%</b>
DistilBERT Fill-in-the-Blank	base	67.03%	66.02%	63.69%	62.67%
RoBERTa Fill-in-the-Blank	base	52.45%	51.11%	33.73%	35.99%
RoBERTa Fill-in-the-Blank	large	51.02%	52.44%	33.14%	34.95%
ALBERT Fill-in-the-Blank	base-v2	31.42%	30.46%	31.84%	31.14%
ALBERT Fill-in-the-Blank	large-v2	31.06%	30.76%	30.08%	33.27%
GA-Reader (baseline)	-	21.23%	21.51%	21.50%	21.86%

Table 1: Results of the Vanilla Fill-in-the-Blank(FitB) Models and GA-Reader

scores. These scores are directly used as logits for the prediction.

### 3.2.5 GSAMN-based Approaches

**BERT-GSAMN-Cloze** Lai et al. (2019) propose a combination of Gated-Attention and Self-Attention - Gated Self-Attention (GSA). They show improvements on smaller datasets compared to Compare-Aggregate Approaches. We use two GSA layers on top of BERT Embeddings, and use the same decoder and selection method as BERT FitB.

## 4 Experimental Setup

In all our experiments, we use the PyTorch implementations of the transformers-based models provided by the HuggingFace (Wolf et al., 2019). The metric for all the 3 subtasks is accuracy. For subtask-I, to obtain the linguistic features mentioned in 3.2, and to obtain the hypernyms and hyponyms for subtask-II, we use the lexical database, WordNet provided by NLTK (Bird and Loper, 2004), a library in Python. For both subtasks, we train our models on train + trial dataset, and evaluate them on the dev set.

The training and the evaluation of systems was on Google Colaboratory’s free GPU (NVIDIA K80/P100). The training time varies with the models. It is around 1-2 hours for the base variants and 2-4 hours for the large models, which is well within the 12 hour limit of Colab. DistilBERT took about half an hour for training.

For finetuning the BERT FitB Hypr Aug Model on the augmented dataset on the MLM task, we use Nvidia-DGX Station with the following specifications: four 32 GB Tesla V100 GPUs, 256 GB RAM and forty Intel Xeon 2.20GHz processors since it is a computationally intensive task.

### 4.1 Hyperparameters

For all our experiments, we use Adam Optimiser (Kingma and Ba, 2017) and Cross Entropy Loss. For choosing the optimal set of hyperparameters, we run a Grid Search on our models. We zero in on a learning rate of 1e-5. Schedulers such as Linear Scheduler, Cosine Annealing Scheduler, etc. seem to have a negative impact on the results. For the FitB models, we keep all the layers unfrozen. Additionally, the maximum input length is kept as 512. We train our models for 4 epochs, keeping a batch size of 2.

### 4.2 Ablation Study/Results

Among the vanilla models, BERT FitB Large performs the best. This is understandable when it comes to DistilBERT and ALBERT, since these models are pruned and distilled for faster computation. Notably, DistilBERT gives comparable performance to BERT FitB Base. A slightly surprising observation was that there is a degradation in accuracy on using RoBERTa. This could be because even though it was pretrained more robustly than BERT on the MLM task, it was not pretrained on the Next Sentence Prediction Task, and hence, might perform worse on Textual Entailment tasks. A peculiar observation is that the large variants of ALBERT FitB and RoBERTa FitB models perform worse than their base variants. This may imply that more training data is needed to train the large variants. For subtask-I, in table 2, we also demonstrate the results of BERT Ensemble, in which we ensemble (i.e., averaging over the predictions) two checkpoints saved during the training process. When it comes to the *Difference Method* using Linguistic Features for imperceptibility, we observe an improvement on the dev set, but a slight fall is observed while evaluating it on the test set. This might be solved by careful tuning of the threshold.

The polls are already years overdue and were scheduled for Sunday . They were postponed because of an ongoing stalemate between the government and a group of opposition senators over an electoral law . Haiti is the poorest country in the region and is still struggling to recover from a 2010 earthquake . Protesters lit piles of wood in the central neighbourhood of Bel Aire before marching to a wealthy hillside neighbourhood , where riot police guarded hotels , shops and Haiti 's elections office . Some demanded President Michel Martelly 's resignation for his " inability to organise elections in the country " . Two opposition activists who had organised the protest were arrested by police for " public unrest and inciting violence " . Mid - term senate elections in Haiti had been due in May 2012 , while the municipal poll is three years behind schedule as Haiti slowly emerges from the earthquake which left much of the country devastated in 2010 . In June , President Michel Martelly decreed that the elections be held on 26 October . The date was set after lengthy talks mediated by the president of Haiti 's Bishops ' Conference , Cardinal Chibly Langlois , intended to overcome the political deadlock between the opposition and the government . But after the National Assembly failed to pass an electoral law in time , the office of Mr Martelly announced another postponement on Sunday . No new date has been set , but the statement said that " President Michel Martelly , in his constant concern to guarantee political stability , promises to pursue consultations with the different sectors of national life in order to hold the elections as soon as possible " . Opposition politicians accuse President Martelly of wanting to rule by decree - a likely scenario if no elections are held before the lower chamber 's term runs out in January . The government argues that opposition politicians are also dragging their feet in the hope of extending their time in office without elections . Thousands of Haitians marched in the capital Port - au - Prince on Sunday in protest at a delay in the country 's [MASK] and municipal elections .

Options: Local, Annual, **Legislative**, Municipal, Devastating

Figure 3: Explanation of a Correctly Classified Sample from Subtask-I (Imperceptibility). The correct option is highlighted in green.

Model	Variant	Val Acc.	Test Acc.
BERT FitB LF	large	75.75%	75.06%
DistilBERT FitB LF	base	68.10%	65.73%
BERT FitB ENS	large	75.15%	<b>77.28%</b>
BERT FitB ENS LF	large	75.87%	75.26%
BERT FitB EM	large	76.58%	76.35%
BERT FitB EM LF	large	<b>76.82%</b>	76.10%
BERT FitB VS	large	76.58%	76.54%
BERT FitB VS LF	large	<b>76.82%</b>	76.20%
BERT MC	large	74.07%	73.76%

Table 2: Results and Ablation Study of the Improvement Methods on Subtask-I<sup>0</sup>

In the future, we aspire to learn embeddings using these Linguistic Features as input to common models such as Word2Vec (Mikolov et al., 2013).

For non-specificity, with the hypernym augmentation method, BERT FitB achieves lower accuracy. A possible reason for this could be that replacing the nouns with their hypernyms in some contexts changes the meaning of the sentence (even though we use Lesk Algorithm for WSD, not all hypernyms make sense). For example, the word "drink" is replaced with "food". For the hyponyms method, we can improve our results by recursively generating hyponyms for a particular option, instead of taking the immediate hyponyms. Again, threshold tuning may help.

In Table 3, a positive sign for the *Difference Method* or the *Threshold Method* is the improvement in the results of BERT FitB Voting (Exact Matching) when we consider the hyponyms. The accuracy jumps from 72.86% to 75.79% on the dev set and from 77.83% to 78.98% on the test set. This reinforces our claim that with more careful tuning of the threshold, we might get improvements on the test set in other methods.

Model	Variant	Val Acc.	Test Acc.
BERT FitB Hypo	large	75.09%	72.83%
BERT FitB Hypr Aug	large	62.26%	60.78%
BERT FitB Hypr Aug Hypo	large	64.51%	55.52%
BERT FitB EM	large	72.86%	77.83%
BERT FitB EM Hypo	large	<b>75.79%</b>	<b>78.98%</b>
BERT FitB VS	large	73.09%	77.59%
BERT FitB VS Hypo	large	75.56%	78.63%
BERT MC	large	71.33%	71.21%

Table 3: Results and Ablation Study of the Improvement Methods on Subtask-II<sup>0</sup>

BERT FitB Voting performs better than vanilla BERT FitB on both subtasks. This is intuitive since in the latter, we truncate the article to 512 tokens without any consideration of how much context is lost. Voting, on the other hand, considers all contexts and hence, gives a superior performance.

For GA-Reader-BERT, when compared with the GA-Reader baseline, the accuracy improves from 21% to 39% on subtask-I dev set. Due to computational restrictions, we couldn't pretrain GA-BERT, and only fine-tuned it for subtask-I to get an idea about its performance, which was sub-optimal (19%). The Answer-Attention system gave us a dev score of  $\approx 61\%$  on subtask-I, which is much higher than the baseline.

BERT-GSAMN-Cloze achieves  $\approx 31\%$  accuracy on subtask-I dev set. The reasons for this could be lack of pretraining, unlike the original paper, or different way to getting the output logits. We see improvement as we reduced number of layers to 1 ( $\approx 38\%$ ) and to 0 ( $\approx 73\%$ ). Hence, we discarded this approach.

<sup>0</sup>LF=Linguistic Features, ENS=Ensemble, FitB=Fill-in-the-Blank, EM=Exact Matching, VS=Voting (similarity), MC=Max Context, Aug=Augmentation, Hypr=Hypernym, Hypo=Hyponym

The Royal College of Physicians of Edinburgh warned that being overweight may now be considered "the norm". It claimed a tax would help fund the "spiralling" healthcare costs associated with the problem. The British Soft Drinks Association (BSDA) insisted that the case is "not compelling". It cited research which suggested a 20% tax would save just four calories per day. Liverpool University chair of clinical epidemiology, Simon Capewell, is due to speak at a conference on the issue in Edinburgh later, entitled: "Obesity: A 21st Century Epidemic". Professor Capewell will cite Mexico as one example where a 10% sugary drinks tax is believed to have contributed to a 10% reduction in the consumption of such beverages while Finland, France, Hungary, Latvia and the USA have also introduced sugar taxes. He said: "The revenues raised can then be invested back into initiatives to increase children's health in these countries, as is happening in Mexico." Scotland has an excellent track record in addressing public health issues. Notable achievements include smoke-free public places and proposals for minimum unit pricing for alcohol. We need to explore how these developments could be repeated with sugary drinks. Gavin Partington, BSDA director general, said: "The efforts by soft drinks companies including product reformulation, smaller pack sizes and increased promotion of low and no-calorie drinks have led to a 7% reduction in calories from soft drinks in the last three years." It's also worth noting that politicians in Belgium and Denmark rejected the notion of a tax in 2013 and the experience in France shows that while sales of soft drinks initially fell after a tax was introduced in 2012, they have increased since. Doctors have called for the introduction of a tax on sugary drinks and drinks to tackle what they describe as an "obesity epidemic".

Options: Food, Terms, Head, Unit, Snacks

Figure 4: Explanation of a Correctly Classified Sample from Subtask-II (Non-Specificity). The correct option is highlighted in green.

Imperceptibility		Non-Specificity	
Model	Test Acc.	Model	Test Acc.
BERT FitB	65.64%	BERT FitB	61.83%
DistilBERT FitB	52.16%	BERT FitB with Hyponyms	59.95%
DistilBERT FitB + Linguistic Features	51.61%	BERT FitB with Hypernym Augmentation	45.98%
BERT FitB + Linguistic Features	65.54%	BERT FitB Voting (Exact Matching)	62.27%
BERT FitB Ensemble + Linguistic Features	64.95%	-	-

Table 4: Submitted Results of Subtask-III: Testing the performance of a system that is trained on one subtask and evaluated on the other.

### 4.3 Analysis of BERT FitB using Integrated Gradients

We use the method of Integrated Gradients (Sundararajan et al., 2017). We follow Ramnath et al. (2020) to compute the word-wise attribution scores for BERT FitB for both subtasks. We compute the Integrated Gradients of the target with respect to the embedding outputs. The Riemann Right Approximation Method with  $n_{steps} = 25$  is used. After obtaining the token-wise attribution scores, we obtain the word-wise attribution scores by using token-to-word offset mapping. We pick the top-10 word-wise attribution scores and normalise them. To implement IG, we use the Captum (Kokhlikyan et al., 2020) library. In favour of brevity, we present one example for each subtask.

In Fig. 3, the correct answer is "legislative". The attribution scores of words like *senate*, *senators*, *municipal* and *President* are high, as is demonstrated by the intensity of the colour. The word "legislative" is, in a sense, more imperceptible than any of the words mentioned above. The *senate* is the legislative branch of the government, and *senators* are its members; *municipal* refers to municipal corporations which are the grassroots governing bodies, etc. Moreover, other words such as *elections*, *political*, *country* also have high attribution scores. These words are related to "legislative"

which exhibits the fact that BERT FitB is not only able to learn the concept of imperceptibility, but is also able to predict a suitable word.

Similarly, in Fig. 4, the correct answer is "food". Note that "snacks" is also an option; however, food is more non-specific than "snacks" and hence, food is the correct option. Another interesting thing to note is the high attribution scores for words/phrases like *calories*, *beverages*, *sugar* and *sugary drinks*. This backs the fact that the model is able to learn the concept of non-specificity, i.e., the above mentioned words are essentially hyponyms of "food".

## 5 Conclusion

We tried out myriad approaches, taking care to not only focus on the architecture aspect, but also how we can quantify imperceptibility and non-specificity. Although we did not achieve favourable improvements in all approaches, we did observe gains in accuracy on the dev set. We reckon that with more careful tuning of parameters such as the threshold in the *Difference Method*, we will be able to achieve these gains on the test set.

We further interpreted the outputs of transformers-based models using Integrated Gradients, and demonstrated that transformer models are able to learn the concepts of imperceptibility and non-specificity. In the future, we intend



to solidify our proposed approaches and carry out further research in this interesting field.

## Acknowledgments

We thank Rajaswa Patil<sup>1</sup> and Somesh Singh<sup>2</sup> for their support. We would also like to express our gratitude to our colleagues at the Language Research Group (LRG)<sup>3</sup>, who have been with us at every stepping stone.

## References

- Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. [SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining](#). In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).
- Steven Bird and Edward Loper. 2004. [NLTK: The natural language toolkit](#). In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.
- Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. [Large-scale Simple Question Answering with Memory Networks](#).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Bhuvan Dhingra, Hanxiao Liu, Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2017a. [Gated-attention readers for text comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1832–1846, Vancouver, Canada. Association for Computational Linguistics.
- Bhuvan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017b. [Gated-Attention Readers for Text Comprehension](#).
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.
- Shi Feng, Zhiqiang Cai, and Danielle McNamara. 2011. Simulating human ratings on word concreteness.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching Machines to Read and Comprehend](#).
- Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A Method for Stochastic Optimization](#).
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. [Captum: A unified and generic model interpretability library for pytorch](#).
- Tuan Lai, Quan Hung Tran, Trung Bui, and Daisuke Kihara. 2019. [A gated self-attention memory network for answer selection](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5953–5959, Hong Kong, China. Association for Computational Linguistics.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [ALBERT: A lite BERT for self-supervised learning of language representations](#). *CoRR*, abs/1909.11942.
- Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient Estimation of Word Representations in Vector Space](#).
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#).
- Sahana Ramnath, Preksha Nema, Deep Sahni, and Mitesh M. Khapra. 2020. [Towards interpreting BERT for reading comprehension based QA](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3236–3242, Online. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter](#). *CoRR*, abs/1910.01108.
- Robert Schreuder and R. Harald Baayen. 1995. Modelling morphological processing. In Laurie B. Feldman, editor, *Morphological aspects of language processing*, pages 131–154. Erlbaum.

<sup>1</sup><https://rajaswa.github.io/>

<sup>2</sup><https://someshsingh22.github.io/>

<sup>3</sup><https://lrg.saidl.in/>

- Otfried Spreen and Rudolph W. Schulz. 1966. [Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns](#). *Journal of Verbal Learning and Verbal Behavior*, 5(5):459 – 468.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. [Axiomatic attribution for deep networks](#). *CoRR*, abs/1703.01365.
- Shinya Tanaka, Adam Jatowt, Makoto Kato, and Katsumi Tanaka. 2013. [Estimating content concreteness for finding comprehensible documents](#). pages 475–484.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *CoRR*, abs/1910.03771.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google’s neural machine translation system: Bridging the gap between human and machine translation](#). *CoRR*, abs/1609.08144.
- Qizhe Xie, Guokun Lai, Zihang Dai, and Eduard Hovy. 2018. [Large-scale cloze test dataset created by teachers](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2344–2356, Brussels, Belgium. Association for Computational Linguistics.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading Comprehension of Abstract Meaning.

# IIE-NLP-Eyas at SemEval-2021 Task 4: Enhancing PLM for ReCAM with Special Tokens, Re-Ranking, Siamese Encoders and Back Translation

Yuqiang Xie Luxi Xing Wei Peng Yue Hu<sup>§</sup>

Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China  
School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China  
{xieyuqiang, xingluxi, pengwei, huyue}@iie.ac.cn

## Abstract

This paper introduces our systems for all three subtasks of SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning. To help our model better represent and understand abstract concepts in natural language, we well-design many simple and effective approaches adapted to the backbone model (RoBERTa). Specifically, we formalize the subtasks into the multiple-choice question answering format and add special tokens to abstract concepts, then, the final prediction of QA is considered as the result of subtasks. Additionally, we employ many finetuning tricks to improve the performance. Experimental results show that our approach gains significant performance compared with the baseline systems. Our system<sup>¶</sup> achieves eighth rank (87.51%) and tenth rank (89.64%) on the official blind test set of subtask 1 and subtask 2 respectively.

## 1 Introduction

The computer’s ability in understanding, representing, and expressing abstract meaning is a fundamental problem towards achieving true natural language understanding. SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning (ReCAM) provides a well-formed benchmark that aims to study the machine’s ability in representing and understanding abstract concepts (Zheng et al., 2021).

The Reading Comprehension of Abstract Meaning (ReCAM) task is divided into three subtasks, including Imperceptibility, Nonspecificity, and Interaction. Please refer to the task description paper (Zheng et al., 2021) for more details. To address the above challenges in ReCAM, we first formalize all subtasks as a type of multiple-choice

Question Answering (QA) task like (Xing et al., 2020). Recently, the large Pre-trained Language Models (PLMs), such as GPT-2 (Radford et al., 2019), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), demonstrate their excellent ability in various natural language understanding tasks (Wang et al., 2018; Zellers et al., 2018, 2019). So, we employ the state-of-the-art PLM, RoBERTa, as our backbone model. Moreover, we design many simple and effective approaches to improve the performance of the backbone model, such as adding special tokens, sentence re-ranking, label smoothing and back translation.

This paper describes approaches for all subtasks developed by the IIE-NLP-Eyas Team (Natural Language Processing group of Institute of Information Engineering of the Chinese Academy of Sciences). Our contributions are summarized as the followings:

- We design many simple and effective approaches to improve the performance of the PLMs on all three subtasks, such as special tokens, sentence re-ranking, siamese encoders and back translation and label smoothing;
- Experiments demonstrate that our proposed methods achieve significant improvements compared with baselines and we obtain the 8th-place in subtask-1 and the 10th-place in subtask-2 on the final official evaluation.

## 2 Approaches

Since the format of the tasks in ReCAM is the same, we use the unified framework to address all tasks. The following is the detail of our methods.

**Task Definition** We first present the description of symbols which are used in this paper. Formally, suppose there are seven key elements in all subtasks, i.e.  $\{D, Q, A_1, A_2, A_3, A_4, A_5\}$ . We sup-

<sup>§</sup>Corresponding author.

<sup>¶</sup>Our Code is publicly available at <https://github.com/indexfziq/IIE-NLP-Eyas-SemEval2021>.

pose the  $D$  denotes the given article, the  $Q$  denotes the summary of the article with a *placeholder*, the  $A_*$  denotes the candidate abstract concepts for all subtasks to fill in the *placeholder*.

**Multi-Choice Based Model** The pre-trained language models have made a great contribution to MRC tasks. Recently, a significant milestone is the BERT (Devlin et al., 2019), which gets new state-of-the-art results on eleven natural language processing tasks. In this section, we present the description of the multi-choice based model which we use in all subtasks. Consider the BERT-style model RoBERTa’s (Liu et al., 2019) stronger performance than BERT, we utilize it as our backbone model, which introduces more data and bigger models for better performance. A multiple-choice based QA model  $\mathcal{M}$  consists of a PLM encoder and a task-specific classification layer which includes a feed-forward neural network  $f(\cdot)$  and a softmax operation. For each pair of question-answer, the calculation of  $\mathcal{M}$  is as follow:

$$score_i = \frac{\exp(f(S_i))}{\sum_{i'} \exp(f(S_{i'}))} \quad (1)$$

$$S_i = \text{PLM}([Q; A_i; D]) \quad (2)$$

where the  $[\cdot]$  is the input constructed according to the instruction of PLMs, and the  $S_*$  is the final hidden state of the first token ( $\langle s \rangle$ ). For more details, we refer to the original work of PLMs (Liu et al., 2019). The candidate answer which owns a higher *score* will be identified as the final prediction. The model  $\mathcal{M}$  is trained end-to-end with the cross-entropy objective function.

**Special Tokens** Considering the great performance of special tokens in entity and relation extraction (Zhong and Chen, 2021), as well as of the prompt template on commonsense reasoning (Xing et al., 2020), we attach special tokens to highlight the semantic representation of candidate abstract concepts in the input layer. To help the PLMs represent and understand the abstract concept (i.e. option word in ReCAM tasks) in textual description (i.e. summary of the article in ReCAM task), we use  $\langle e \rangle$  and  $\langle /e \rangle$  to add on both ends of the abstract concept, i.e.  $\langle e \rangle$  abstract concept  $\langle /e \rangle$ . It is interesting that the special tokens are useful features contributing to most of the system’s boost, and we have tried many other useful special tokens which will be discussed in section 4.

**Sentence Ranking** As the given passage is too long to be deal with the Pre-trained Language Models (PLMs), we consider refining the passage input by rearranging the order of the sentences in the passage. With this reorder process, the sentence, which is more critical to the question, can appear at the beginning of the passage. Although the passage’s sequential information is sacrificed, we keep the more question-relevant information of the passage. Supposing the passage  $D$  contains  $N$  sentences, i.e.,  $D = \{W_1, W_2, \dots, W_N\}$ , where each sentence  $W_n = \{t_1, t_2, \dots, t_M\}$  including  $M$  tokens. We denote the given cloze-style question as  $Q$ . To rank the sentences in  $D$ , we resort BERT to compute the similarity score between each sentence, i.e.  $W_n$ , and  $Q$  following the algorithm in Zhang et al. (2020). After ranking, the sentences in  $D$  are sorted in descending order of similarity scores, and we can get a rearranged passage  $\hat{D}$  as the passage input to the QA model. In the implement progress,  $\hat{D}$  will be truncated to fit into the PLM encoder with our setting max length.

**Siamese Encoders** When exploring the dataset, we find that the complete question statement, representing the result statement after replacing the *placeholder* token with the candidate option, also contains the semantic information which can help to make the judgment about options. Based on the observation, we propose a siamese encoders based architecture to inject the additional complete question statement information while not influence the input with passage. On the other hand, it can be seen as introducing an auxiliary task to assist the main task. Specifically, the training of siamese encoder based architecture is as following:

$$l_i^1 = \text{PLM}([\hat{Q}_i])[0] \quad (3)$$

$$l_i^2 = \text{PLM}([Q; A_i; D])[0] \quad (4)$$

$$P^1(A_i|\hat{Q}) = \text{softmax}(f(l_i^1)) \quad (5)$$

$$P^2(A_i|D, Q) = \text{softmax}(f(l_i^2)) \quad (6)$$

where the  $\text{PLM}(\cdot)$  stands for PLM encoder,  $\hat{Q}_i$  is the complete question statement,  $i$  indicates the  $i$ -th candidate answer,  $f(\cdot)$  is the feed forward network. To coordinate the two losses, we opt for an uncertainty loss (Kendall et al., 2018) to adjust it adaptively through  $\sigma_{\{1,2\}}$  as:  $\mathcal{L}(\theta, \sigma_1, \sigma_2) = \frac{1}{2\sigma_1^2} \mathcal{L}^1(\theta) + \frac{1}{2\sigma_2^2} \mathcal{L}^2(\theta) + \log \sigma_1^2 \sigma_2^2$ , where  $\mathcal{L}^{\{1,2\}}$  are the cross-entropy loss between the model prediction  $P^{\{1,2\}}$  and the ground truth label respectively.

**Back Translation** Generally speaking, more successful neural networks require a large number of parameters, often in the millions. In order to make the neural network implements correctly, a lot of data is needed for training, but in actual situations, there is not as much data as we thought. The role of data augmentation includes two aspects. One is to increase the amount of training data and improve the generalization ability of the model. The other is to increase the noise data and improve the robustness of the model. A large number of the works (Buslaev et al., 2018; Bloice et al., 2019; Chen et al., 2020; Cubuk et al., 2020; Sato et al., 2018; Zhu et al., 2020) consider the data augmentation to make better performances. In the field of computer vision, a lot of work (Buslaev et al., 2018; Bloice et al., 2019; Chen et al., 2020; Cubuk et al., 2020) uses existing data to perform operations, such as flipping, translation or rotation, to create more data, so that neural networks have better generalization effects. Adding Gaussian distribution to text processing (Sato et al., 2018) can also achieve the effect of data augmentation. Besides, some works (Miyato et al., 2017; Zhu et al., 2020) utilize the adversarial training methods to do the data augmentation. For convenience and simplicity, we adopt the back translation (Sennrich et al., 2016) to increase the amount of training data, which is used to construct pseudo parallel corpus in unsupervised machine translation (Lample et al., 2018). Specifically, we use the Google API<sup>†</sup> to translate the passage into French, and then translate the translation into English in turn. The pseudo parallel corpus can be obtained as:

$$\{D'\} = bkt(\{D\}) \quad (7)$$

where  $\{D'\}$  means the translated English corpus that we used as data agument,  $bkt$  is back translation.

As for the question, given the existence of the special character *placeholder*, forced translation may result in grammatical errors and semantic gaps. Therefore, the questions and options will be kept original. After getting the pseudo parallel corpus, we train our model with the training data together with the cross-entropy loss function.

**Label Smoothing** Furthermore, for improving the generalization ability of the model trained on sole task and prevent the overconfidence of model,

<sup>†</sup>The web page is available at <https://translate.google.com>

Subtask	Train	Trail	Dev	Test
Imperceptibility	3227	1000	837	2025
Nonspecificity	3318	1000	851	2017

Table 1: Data scale of each subtask.

Hyper-parameter	Value
LR	{1e-5, 2e-5}
Batch size	{16, 32}
Gradient norm	1.0
Warm-up	{0.1, 1, 2}
Max. input length (# subwords)	200
Epochs	[3, 10]

Table 2: Hyper-parameters of our approach.

we consider training model with label smoothing (Miller et al., 1996; Pereyra et al., 2017). Label smoothing can maintain uncertainty over the label space during training. When training with label smoothing, for classification tasks, the hard one-hot label distribution is replaced with a softened label distribution through a smoothing value  $\alpha$ , which is a hyperparameter. Specifically, for hard one-hot label distribution, the target category’s probability will be assigned to 1.0 and others are 0.0. Label smoothing will soften the label distribution by modifying the probability distribution with a discount. Then, the target category’s probability will be  $1 - \alpha$ , and the probabilities of the rest categories are  $\frac{\alpha}{\mathcal{K}-1}$ , where  $\mathcal{K}$  is the number of task categories. In our experiments, we set the smoothing value  $\alpha = 0.1$ .

### 3 Experiments and Results

#### 3.1 Experimental Setup

In all subtasks, the scale of each task is shown in Table 1. We train the model on training data and the related pseudo data generated by back translation, then select hyper-parameters based on the best performing model on the dev set, and then report results on the test set.

Our system is implemented with PyTorch and we use the PyTorch version of the pre-trained language models<sup>‡</sup>. We employ RoBERTa (Liu et al., 2019) large model as our PLM encoder in Equation 2. The Adam optimizer (Kingma and Ba, 2014) is used to fine-tune the model. We introduce the detailed setup of the best model on the development dataset. For subtask-1 and subtask-2, the hyper-parameters are shown in Table 2.

<sup>‡</sup><https://github.com/huggingface/transformers>

Models	Trial Acc.	Dev Acc.
ROBERTA <sub>LARGE</sub> (Liu et al., 2019)	85.85	82.12
(1) w/ special tokens	<b>87.81</b>	<b>87.69</b>
(2) w/ sentence ranking	86.54	83.52
(3) w/ label smoothing	86.88	85.85
(4) w/ siamese encoders	86.62	83.22
(5) w/ back translation	87.23	84.32
Our Approach	<b>87.81</b>	<b>87.69</b>

Table 3: The results of our system on subtask-1. Our approach is the final, stable and best model: ROBERTA<sub>LARGE</sub> with special tokens. We finally obtain 87.51 Acc. on the official blind test set.

Models	Trial Acc.	Dev Acc.
ROBERTA <sub>LARGE</sub> (Liu et al., 2019)	<b>88.51</b>	85.93
(1) w/ special tokens	87.47	88.98
(2) w/ sentence ranking	87.29	86.84
(3) w/ label smoothing	87.67	87.08
(4) w/ siamese encoders	87.34	86.18
(5) w/ back translation	88.41	87.54
Our Approach	87.10	<b>89.54</b>

Table 4: The results of our system on subtask-2. Our approach is the final, stable and best model: ROBERTA<sub>LARGE</sub> with special tokens and label smoothing. We finally obtain 89.64 Acc. on the official blind test set.

### 3.2 Evaluation Results

**Imperceptibility** From Table 3, we can see the results of our system on subtask-1 of ReCAM. Compared to the backbone model RoBERTa large model, our methods achieve significant improvements. It is interesting that the special token is the most helpful part for the Imperceptibility subtask.

**Nonspecificity** Table 4 summarizes the results of our approaches on subtask-2 of ReCAM. In Nonspecificity subtask, the model with special tokens and label smoothing performs best. Compared to the backbone model ROBERTA<sub>LARGE</sub>, all our methods achieve better performance.

**Interaction** We also perform subtask-3 of ReCAM, Interaction, which aims to provide more insights into the relationship of the two views on abstractness. In this task, we test the performance of our system that is trained on one definition and evaluated on the other. The results of our system’s performance on Imperceptibility and Nonspecificity subtasks which is shown in Table 5. We can find that our model is relatively robust for different abstract concepts.

Trained on	Tested on	Test Acc.
Subtask-1	Subtask-1	87.51
Subtask-1	Subtask-2	84.13
Subtask-2	Subtask-2	89.64
Subtask-2	Subtask-1	81.09

Table 5: The results of our approach on subtask-3.

Special Token	Trial Acc.	Dev Acc.
<e> </e>	88.01	<b>87.10</b>
<#> </#>	<b>88.63</b>	86.93
<\$> </\$>	88.12	86.26
# /#	87.34	85.89
\$ /\$	87.73	86.13
N/A	86.23	83.12

Table 6: The results of models with different special tokens on subtask-1.

## 4 Analysis and Discussion

### 4.1 Ablation Study

In this part, we perform an ablation study of our approaches (special tokens, sentence re-ranking, label smoothing, siamese encoders and back translation).

Table 3 and 4 shows that our proposed methods help the backbone model better represent and understand the abstract concepts. Note that the special tokens bring the PLMs with the best improvements in both subtask-1 and subtask-2. It is possible that the special tokens teach the model to focus on the abstract concept in a stronger manner. Moreover, other common tricks bring with little improvements.

### 4.2 Discussion of Special Tokens

We also search for the best special tokens for ReCAM on the dev set of subtask-1. `e` stands for the word `entity`. `#` and `$` are common special tokens for NLP downstream applications.

As shown in Table 6, `<e> </e>` enhance the representations of abstract concepts best of all. `#` and `$` work well. In addition, the `<>` and `</>` could be helpful for PLMs to pay attention to the abstract concepts. Moreover, it is interesting that each special token helps PLMs choose the right abstract concepts which are submerged in long sequential tokens (including article and summary). This result strengthen the point that special tokens can enhance the representation of abstract concepts in PLM based approaches.

## 5 Conclusion

In this paper, we design many simple and effective approaches to improve the performance of the PLMs on all three subtasks. Experiments demonstrate that the proposed methods achieve significant improvement compared with the PLMs baseline and we obtain the eighth-place in subtask-1 and tenth-place in subtask-2 on the final official evaluation. Moreover, we show that special tokens are useful features contributing to most of the system’s boost, which work well in enhancing PLMs for representing and understanding abstract concepts.

## Acknowledgements

We thank the anonymous reviewers for their insightful feedback. This work has been supported by the National Key Research and Development Program of China under Grant No.Y750211101.

## References

- Marcus D. Bloice, Peter M. Roth, and Andreas Holzinger. 2019. [Biomedical image augmentation using augmentor](#). *Bioinform.*, 35(21):4522–4524.
- Alexander V. Buslaev, Alex Parinov, Eugene Khvedchenya, Vladimir I. Iglovikov, and Alexandr A. Kalinin. 2018. [Albumentations: fast and flexible image augmentations](#). *CoRR*, abs/1809.06839.
- Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. 2020. [Gridmask data augmentation](#). *CoRR*, abs/2001.04086.
- Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. 2020. [Randaugment: Practical automated data augmentation with a reduced search space](#). In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*, pages 3008–3017.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *NAACL-HLT*, pages 4171–4186.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. [Multi-task learning using uncertainty to weigh losses for scene geometry and semantics](#). In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7482–7491.
- Diederik P. Kingma and Jimmy Ba. 2014. [Adam: A method for stochastic optimization](#). *arXiv*.
- Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc’Aurelio Ranzato. 2018. [Unsupervised machine translation using monolingual corpora only](#). In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- David J. Miller, Ajit V. Rao, Kenneth Rose, and Allen Gersho. 1996. [A global optimization technique for statistical classifier design](#). *IEEE Trans. Signal Process.*, 44(12):3108–3122.
- Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. [Adversarial training methods for semi-supervised text classification](#). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.
- Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. [Regularizing neural networks by penalizing confident output distributions](#). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#). In *OpenAI Blog*.
- Motoki Sato, Jun Suzuki, Hiroyuki Shindo, and Yuji Matsumoto. 2018. [Interpretable adversarial perturbation in input embedding space for text](#). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4323–4330.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Improving neural machine translation models with monolingual data](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Luxi Xing, Yuqiang Xie, Yue Hu, and Wei Peng. 2020. [IIE-NLP-NUT at SemEval-2020 task 4: Guiding plm with prompt template reconstruction strategy for comve](#). In *SemEval@COLING*.

- Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. [SWAG: A large-scale adversarial dataset for grounded commonsense inference](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. [HellaSwag: Can a machine really finish your sentence?](#) In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.
- Tianyi Zhang, Varsha Kishore\*, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [BERTScore: Evaluating text generation with BERT](#). In *International Conference on Learning Representations*.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Zexuan Zhong and Danqi Chen. 2021. [A frustratingly easy approach for entity and relation extraction](#). In *North American Association for Computational Linguistics (NAACL)*.
- Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. [FreeLB: Enhanced adversarial training for natural language understanding](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.



# NLP-IIS@UT at SemEval-2021 Task 4: Machine Reading Comprehension using the Long Document Transformer

**Hossein Basafa**  
College of ECE  
University of Tehran  
hbasafa@ut.ac.ir

**Sajad Movahedi**  
College of ECE  
University of Tehran  
s.movahedi@ut.ac.ir

**Ali Ebrahimi**  
College of ECE  
University of Tehran  
ali96ebrahimi@ut.ac.ir

**Azadeh Shakery**  
College of ECE  
University of Tehran  
shakery@ut.ac.ir

**Heshaam Faili**  
College of ECE  
University of Tehran  
hfaili@ut.ac.ir

## Abstract

This paper presents a technical report of our submission to the 4th task of SemEval-2021, titled: Reading Comprehension of Abstract Meaning. In this task, we want to predict the correct answer based on a question given a context. Usually, contexts are very lengthy and require a large receptive field from the model. Thus, common contextualized language models like BERT miss fine representation and performance due to the limited capacity of the input tokens. To tackle this problem, we used the longformer model to better process the sequences. Furthermore, we utilized the method proposed in the longformer benchmark on wikipop dataset which improved the accuracy on our task data from (23.01% and 22.95%) achieved by the baselines for subtask 1 and 2, respectively, to (70.30% and 64.38%).

## 1 Introduction

Reading comprehension is the ability to understand a passage either by human or machine. One of the great benchmarks to evaluate this ability is to try to answer specific questions related to the passage (Rajpurkar et al., 2016). Generally, this problem can contain single or multiple documents as context (containing relevant information needed to understand and answer the question), a question (a sentence with at least one asking parameter), and an answer (which is the parameter value of the question).

In the Task of Reading Comprehension of Abstract Meaning (ReCAM), we have one passage as a context, one question and five candidate answers (Zheng et al., 2021). The goal is to identify the correct answer based on the context and the given question. You can see a sample of the data in Table 1. For each instance of the data, there is a passage, a question with a missing word that should be filled based on the passage, and five candidate answers to the question.

<b>Passage</b>	... observers have even named it after him, “Abenomics”. It is based on three key pillars - the “three arrows” of monetary policy, fiscal stimulus and structural reforms in order to ensure long-term sustainable growth in the world’s third-largest economy. In this weekend’s upper house elections ...
<b>Question</b>	Abenomics: The <i>@Placeholder</i> and the risks
<b>Answer</b>	(A) chances (B) prospective (C) security (D) objectives (E) threats

Table 1: An instance of the data.

The task divides into two subtasks: imperceptibility and non-specificity (Zheng et al., 2021).

- imperceptibility: this level of abstract words refers to ideas and concepts that are distant from immediate perception; such as culture, economics, and politics.
- non-specificity: In contrast to concrete words, this subtask includes more abstract words which focus on a different type of definition; for example, a concrete word like ‘cow’ could be interpreted as an ‘animal’ which is considered as a more abstract word (Changizi, 2008).

The main challenges of this task are the abstract meaning concept representation as well as the machine reading comprehension. This is the main reason we have utilized contextualized language representation models to tackle abstract meaning representation problems.

In this paper, we use an end-to-end deep contextualized architecture to model this task. This model is also capable of considering more than

one passage as the context, and more than five candidate answers. Since we use the long document transformer model (Longformer (Beltagy et al., 2020)), no limitation is considered in context passage length. We have evaluated this model both on subtask-1 and subtask-2 which resulted in 70% and 64% accuracy, respectively. Therefore, we have about 40% improvement compared to the baseline, which is a Gated Attention (GA) model (Zheng et al., 2021).

The rest of the paper is as follows: Section 2 describes the related works and the background. Section 3 includes the description of the proposed method. Section 4 contains the evaluation metrics used as well as a brief discussion, which is then followed by a conclusion and future works in section 5.

## 2 Background and Related Works

Many approaches have been presented in the literature, from pipeline-based models to end-to-end ones. Each module is also well-investigated from rule-based models to deep learning ones. Despite various configurations presented in the literature to model this problem, most of the systems consist of three modules (Baradaran et al., 2020):

- Language representation: this module is responsible to encode the inputs. Context, question, and answer need to be represented as numeric values for computational algorithms to be usable on them. Dense vectorized representations are the most popular methods, which allow us to use the majority of machine learning algorithms.
- Reasoning: this module is used to find demonstrations of why the answer is assumed to be valid. It can also be used as a limiter for searchable context.
- Prediction: this module aims to generate, retrieve or select the correct answer based on the task description.

Recent studies are provided as follows with respect to these modules that the last two modules have been merged. In the end, the longformer model is presented as our mainstay in this paper.

### 2.1 Word and text representation

One of the most important problems in NLP is representation learning. The earliest models for word

representation in the time of deep learning were the models proposed in (Pennington et al., 2014) and (Mikolov et al., 2013), which utilized the weights learned for an auxiliary task (a simplified version of the task of language modeling) for word representation. Similarly, methods proposed in (Le and Mikolov, 2014) and (Liu et al., 2015) utilized a similar structure for sentence, paragraph, or document representation learning.

While these methods were quite effective, it has been shown that using neural language models as a way of word representation results in much better, and context-aware representations. In (Howard and Ruder, 2018) it has been shown that fine-tuning language models as sentence encoders result in a significant performance improvement. At the same time, (Peters et al., 2018) used language models directly as word representations, which resulted in significant improvements. In (Devlin et al., 2018) a transformer model was trained for the task of masked language models, which resulted in significant improvements, surpassing human performance in many NLP tasks. One of the shortcomings of transformers is the lack of a memory mechanism, which results in (theoretically) lower receptive field compared with LSTMs (Beltagy et al., 2020) this shortcoming was addressed by improving the self attention mechanism in transformers so that it would have a (theoretically) unbounded receptive field. More details are presented later in this section.

### 2.2 Natural language understanding

Natural language understanding (NLU) is an umbrella term, referring to any tasks that require machine comprehension. Compared to other NLP tasks, NLU requires the model to be able to understand and reason about the data (Semaan, 2012). While great progress has been made in this field by using contextual word representation (Devlin et al., 2018), it has been found that designing the model itself must not be neglected (Zhu et al., 2018). On the other hand, it has been shown that utilizing a transfer learning setting to share knowledge between different NLU tasks results in better performance with fewer data and fewer parameters (Pilault et al., 2020), which proves a significant similarity between these tasks.

### 2.3 The Longformer

Deep contextualized language models like BERT (Devlin et al., 2019) have been well investi-

gated in the literature and achieved state-of-the-art results on various tasks. However, these models suffer from performance limitations due to their self-attention layer which results in quadratic space and time complexity concerning the sequence length. In contrast, this model removes the self-attention layer from the base language models, so the limitation resolves and the complexity scales to linear. In order to increase the quality of the model compared to basic models, they have added a global attention layer to the model end which significantly outperforms state-of-the-art models on long document (passage) tasks and competitive on normal documents. Also, this configuration increases the performance on both normal and lengthy inputs which makes it a good alternative for tasks containing large inputs. This model is also evaluated on a similar task on WikiHop dataset(Welbl et al., 2018) and improved the results in terms of accuracy(Beltagy et al., 2020).

### 3 Method

As mentioned in section 1, given a passage, a question, and a set of answers to the question, the goal is to predict the correct answer among the candidates, which can be seen as a benchmark to evaluate how well the model can comprehend the abstract meaning. To do so, we considered an end-to-end deep learning architecture based on the transformer architecture.

Specifically, we used contextual word embeddings based on the transformer to better discover and encode the information contained in the passage. In our model, both subtasks use the same architecture as shown in figure 1, although we did not experiment on the possibility of multi-task learning. The word representation models are fine-tuned on the data for better performance. The fine-tuning procedure could allow us to extract additional, task-related information which could result in better accuracy in the evaluation phase.

To model this problem, let  $c = \{c_1, c_2, \dots, c_I\}$  denote the passage as the context, where  $c_i$  corresponds to the  $i^{th}$  token (word or subword, depending on the tokenization technique used) and  $I$  is the number of tokens in the passage. Similarly, the question is considered as  $q = \{q_1, q_2, \dots, q_K\}$  where  $K$  denotes the length of the question, and  $q_k$  corresponds to the  $k^{th}$  token of the question. Each answer also denotes as  $e^j$  which is only one abstract word ( $j \in \{1, 2, \dots, 5\}$ ). Then we concatenate

the question and the candidates as:

$$a = [q; e^1; e^2; \dots; e^5]. \quad (1)$$

The size of this sequence is  $A = K + 5$  as we have only 5 candidates. Generally, this can be an arbitrary length based on the dataset.

Note that we introduce special tokens to separate the context, the question, and the candidates, similar to (Beltagy et al., 2020). Specifically, we introduce the tokens  $\langle s \rangle$  and  $\langle /s \rangle$  for separating the context,  $\langle q \rangle$  and  $\langle /q \rangle$  for separating the the question, and the tokens  $\langle ent \rangle$  and  $\langle /ent \rangle$  for separating the candidates from each other. In the case of multiple passages, all passages are concatenated to form a single context. These tokens are randomly initialized and fine-tuned.

We used the Longformer model introduced in (Beltagy et al., 2020) as the pre-trained contextual embedding model in our method. Since the context could be too long, we split the context sequence to separate chunks. Each chunk length is equal to maximum sequence length the model could accept appending the sequence  $a$ ; in fact,  $model\_max\_length = len(chunk) + len(a)$ . If  $c^l$  denote each chunk, this sequence could be showed as:

$$b = [c^l; a] \quad (2)$$

where the full context is  $c = \{c^1, c^2, \dots, c^L\}$ , and  $L$  is the last chunk. The size of this sequence is  $B$  so  $B = L + A$ .

After feeding the input  $b$  to the Longformer model, we apply a global attention only on  $a$  (concatenated question and answer candidates), and the rest is the context. As the longformer model utilizes a base model (like RoBERTa without the self-attention layer, in our case), we denote this as *basemodel* function that outputs the encoded sequence of the input. If  $GAttn$  denotes the global attention function, we have:

$$d_i = basemodel(b) \quad (3)$$

$$g_i = GAttn(d_i).1(i \in A) \quad (4)$$

where  $d_i$  is the raw output vector for each input token. The global attention function is applied if it is a question or answer candidate token. Then, we extract the outputs corresponding to the question and the candidates tokens, i.e. we have:

$$h_j = GAttn(a, c^l) \quad (5)$$

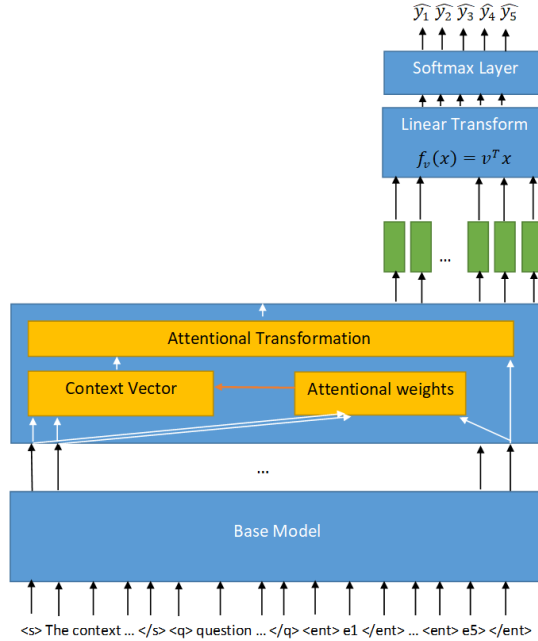


Figure 1: The model architecture. The concatenated input vector will be encoded using the base model (like RoBERTa without the self-attention layer, in our case). A global attention(Luong et al., 2015) will be applied to the question and the candidate answers representations with respect to the passage as the context. The logit (score) of each ent token will be calculated using a linear transformation function, then the prediction distribution over the answer candidates (ent tokens) will be outputted using a softmax layer.

Finally, we obtain the logit of each candidate (<ent> tokens) as  $x_j$  ( $x_j = h_j$  if  $j$  correspond to a candidate), average over different chunks, and apply a linear transformation:

$$f_j = v^T x_j \quad (6)$$

where the vector  $v$  is trainable, and  $f_j$  is the score of each candidate. And the probability distribution over the candidates will be calculated using a softmax layer on the logits. The predicted answer is the argmax of the softmax output. we fine-tuned the model using the cross-entropy loss.

## 4 Evaluation

Although we only participated in the second subtask, we will evaluate our model on both subtasks here. We will explain our configurations for utilizing the model on the task as well as other baselines which are the BERT-base as an alternative model and the Gate-Attention (GA) as our task baseline. Finally, a brief discussion will be done based on the results.

### 4.1 Metrics

Popular metrics to evaluate these models are F1, EM (Exact Match or accuracy), and MRR (Mean

Reciprocal Rank). As the precision and recall in our task are equal, so  $F1 = \text{Precision} = \text{Recall}$ . Also, F1 and EM are the same. And, the use of MRR is optional, so the metrics used to evaluate the result are the accuracy and the F1.

### 4.2 Baseline configuration

The baseline model (GA) is trained for 30 epochs, each epoch containing 101 mini-batches. The train batch size is set to 32. Dropout with the rate of 0.5 is also applied to the hidden states, and the learning rate is set to 0.001. The dimensionality of the GloVe embedding is 300, and the hidden size is set to 128. Training and evaluation take about 2 hours on a single v100 GPU.

### 4.3 BERT configuration

We use the same configuration as our method except for the global attention mechanism. In fact, we consider the output vector of each chunk as our final vector to be linearly transformed into single logit, followed by a softmax layer using the cross-entropy loss. Similarly, the logit is averaged over different chunks, before applying the linear transformation. Note that the maximum sequence length here is bounded to 512 tokens, and the model includes the  $n^2$  attention mechanism. We use the

Metrics	Baseline(GA)	BERT	Our Method
Accuracy	23.01%	63.43%	<b>70.30%</b>
Macro Avg F1	22.83%	63.38%	<b>70.23%</b>
Weighted Avg F1	22.76%	63.40%	<b>70.27%</b>

Table 2: Subtask1 evaluation metrics on the test set

Metrics	Baseline(GA)	BERT	Our Method
Accuracy	22.95%	58.76%	<b>64.38%</b>
Macro Avg F1	22.42%	58.72%	<b>64.35%</b>
Weighted Avg F1	22.45%	58.75%	<b>64.40%</b>

Table 3: Subtask-2 evaluation metrics on the test set

base version of the model and fine-tuned it on each subtask.

#### 4.4 Our method configuration

We used the same model introduced in section 3 for both subtasks. The model was initialized using the Longformer-base pre-training weights, then fine-tuned in each of the subtasks. Due to the performance issues, the model max sequence length is set to 4096 tokens which are sufficient in our case. We also used the RoBERTa-large tokenizer to tokenize the input sequence as the Longformer model has been trained on using this configuration. We used a batch size of 32 and a maximum learning rate of  $3e-5$  using the Adam optimizer with  $\beta_2=0.98$ . We then assumed the validation check interval to 250 which indicates the number of gradient updates between checking validation loss. And a weight decay of 0.01 has been considered to regularize the model and avoid overfitting.

Our proposed model is trained for 15 epochs for each task. Fine-tuning the model takes about six hours, and inference takes about nine seconds for each sample on a single V100 GPU.

#### 4.5 Evaluation od Subtask 1

Subtask1 measures imperceptibility abstract level of language understanding. This subtask includes 3227 training samples, 837 validation samples, and 2025 test samples. The size of the biggest sample in terms of context length is about 2000 tokens. We have achieved an accuracy of 70% on the validation set, which improves our baseline by about 40 percent. Table 2 showed the results of this subtask.

#### 4.6 Evaluation on Subtask 2

Subtask2 measures the non-specificity level of abstract meaning in reading comprehension. It in-

cludes 3318 training samples, 851 validation samples, and 2017 test samples. The best accuracy on the validation set is 64%. Table 3 showed the results of this subtask.

#### 4.7 Discussion

We used two baselines to find out the effect of using a pre-trained model rather than a simple RNN model. Although this task offers a higher level of representation, using the pre-train models is helpful, and there is a higher chance of modeling such abstract concepts.

The results on subtask2 are weaker than subtask1 in pre-trained models. This can be the consequence of limited semantic representation for abstract word which indicates the subtask2 includes more abstract words in terms of abstract level; for example, the word 'animal' could be matched to any animal, like 'cat' or 'dog', but the word 'entity' is hard to be represented as it could be matched to a large number of words. And the model faces a limitation in the knowledge representation. Another assumption could be the data enrichment that these model has been trained on. As most of the available texts for training consist of concrete words, it is more likely to leverage the language understanding to less abstract words to achieve a better result.

Comparing our method which is based on longformer model to usual language models like BERT indicates a new insight in terms of passage length and the attention mechanism. Popular language models like BERT and RoBERTa use a  $n^2$  attention which requires a large receptive field to represent long passages. This results in the performance limitation which bounds the input sequence up to 512 tokens. In contrast, the longformer global attention mechanism relaxes this limitation as we only need to pay attention to a small factor of context

and more focus on the local window. So the receptive field will not overflow and saves the necessary information to better represent the language.

We have analyzed the errors that mostly affect our model performance. We think that the problem is the contextual representation of the language modeling, which is not well-suited in our method i.e. concatenating the context, question, and answer. The main disadvantage of concatenating the candidate answers to each other is the missing fine contextual representation as the state-of-the-art models consume the position embedding. Additionally, incorrect candidates register additional noise to each word representation as well as the placeholder in the question.

## 5 Conclusion and Future works

We have shown how different approaches can be leveraged to machine reading comprehension of abstract meaning. We reformulated the longformer model to learn abstract meaning as a new level of semantic in machine reading comprehension. This method can also be improved by taking advantage of external knowledge and task-specific model architectures that optimize the current baseline.

## References

- Razieh Baradaran, Razieh Ghiasi, and Hossein Amirkhani. 2020. [A survey on machine reading comprehension systems](#). *CoRR*, abs/2001.01582.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Mark A Changizi. 2008. Economically organized hierarchies in wordnet and the oxford english dictionary. *Cognitive Systems Research*, 9(3):214–228.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR.
- Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29.
- Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Jonathan Pilault, Amine Elhattami, and Christopher Pal. 2020. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *arXiv preprint arXiv:2009.09139*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [Squad: 100, 000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.
- Paul Semaan. 2012. Natural language generation: an overview. *J Comput Sci Res*, 1(3):50–57.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *Transactions of the Association for Computational Linguistics*, 6:287–302.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Chenguang Zhu, Michael Zeng, and Xuedong Huang. 2018. Sdnet: Contextualized attention-based deep network for conversational question answering. *arXiv preprint arXiv:1812.03593*.

# IITK@Detox at SemEval-2021 Task 5: Semi-Supervised Learning and Dice Loss for Toxic Spans Detection

Archit Bansal    Abhay Kaushik    Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{architb, kabhay}@iitk.ac.in

ashutoshm@cse.iitk.ac.in

## Abstract

In this work, we present our approach and findings for SemEval-2021 Task 5 - Toxic Spans Detection. The task’s main aim was to identify spans to which a given text’s toxicity could be attributed. The task is challenging mainly due to two constraints: the small training dataset and imbalanced class distribution. Our paper investigates two techniques, semi-supervised learning and learning with Self-Adjusting Dice Loss, for tackling these challenges. Our submitted system (ranked ninth on the leader board) consisted of an ensemble of various pre-trained Transformer Language Models trained using either of the above-proposed techniques.

## 1 Introduction

Content moderation has become the topic of most conversations regarding social media platforms. However, with over 4 billion active internet users, it is impossible to moderate each piece of message generated online manually. Therefore, the focus is now shifting towards tackling the issue using machine learning methods.

Various toxicity detection datasets (Wulczyn et al., 2017; Borkan et al., 2019) and models (Pavlopoulos et al., 2017; Liu et al., 2019; Seganti et al., 2019) have been successfully developed over the years to tackle the issue of moderation. However, these have mostly focused on identifying whole comments or documents as either toxic or not. In semi-automated settings, a model merely generating a toxicity score for each comment, some of which can be very lengthy, is not of much help to human moderators. To tackle this issue, the *SemEval 2021 Task 5 : Toxic Spans Detection* is introduced (Pavlopoulos et al., 2021). The task involves identifying text spans in a given toxic post that contributes towards the toxicity of that post. The task aims to promote the development of a system that

would augment human moderators by giving them more insights into what actually contributes to the text’s toxicity.

The task is challenging mainly due to the following reasons: a) small size of the dataset b) characteristics of text samples extracted from social media leading to difficulties such as out-of-vocabulary words and ungrammatical sentences c) class imbalance in the dataset d) inconsistencies in data annotations. We approached this task as a sub-token level sequence labeling task. Fine-tuned pre-trained transformer language models (Qiu et al., 2020) are the backbone of all our approaches. We investigated two main techniques to enhance the results of the fine-tuned transformer models, namely Semi-Supervised Learning (Yarowsky, 1995; Liu et al., 2011) and fine-tuning with Self-Adjusting Dice Loss (Li et al., 2020). This paper reports the results of our experiments with these different techniques and pre-trained transformer models. Our submitted system consisted of an ensemble of different pre-trained transformer models and achieved an F1 score of 0.6895 on the test set and secured 9th position on the task leaderboard. All of our code is made publicly available on Github<sup>1</sup>.

The rest of this paper is organized as follows. Section 2 discusses the previous works in the fields of offensive language detection and span identification. Section 3 describes the dataset. Section 4 explains the proposed approaches. Section 5 reports the results of various experiments with the proposed approaches, and section 6 analyzes the proposed approaches via ablation studies. We conclude with an error analysis of our model performance in section 7 and concluding remarks in section 8.

---

<sup>1</sup>[https://github.com/architb1703/Toxic\\_Span](https://github.com/architb1703/Toxic_Span)

## 2 Related Work

As the task involves detecting toxic spans in a text, we present the related work in two parts: (i) Offensive Language Detection and (ii) Span Identification.

**Offensive Language Detection:** Research work has been done on different abusive and offensive language identification problems, ranging from aggression (Kumar et al., 2018) to hate speech (Davidson et al., 2017), toxic comments (Saif et al., 2018), and offensive language (Laud et al., 2020; Pitsilis et al., 2018). Recent contributions to offensive language detection came from the SemEval-2019 Task 6 OffensEval (Zampieri et al., 2019). The task organizers concluded that most top-performing teams either used BERT (Liu et al., 2019) or an ensemble model to achieve SOTA results. Interestingly, the task of locating toxic spans is relatively novel, and its successful completion can be groundbreaking. A recent approach with a narrower scope is by Mathew et al. (2020), who focused on the rationality of decision in the task of hate speech detection.

**Span Identification:** Span detection/identification tasks include numerous tasks like named entity recognition (NER) (Nadeau and Sekine, 2007), chunking (Sang and Buchholz, 2000) and keyphrase detection (Augenstein et al., 2017). (Papay et al., 2020) analyzed the span identification tasks via performance prediction over various neural architectures and showed that the presence of BERT component in the model is the highest positive predictor for these tasks. Inspired by this observation, we have built our model based on the transformer architecture, further exploiting the benefits of semi-supervised learning and modified Dice Loss.

## 3 Dataset

### 3.1 Data Description

The competition dataset comprises around 10K comments extracted from the Civil Comments Dataset and annotated using crowd-raters. The organizers released the dataset in 3 phases: trial, train, and test. The trial dataset consisted of 690 texts, whereas the training dataset consisted of 7939 texts. Moreover, the test set on which our system was finally evaluated consisted of 2000 text samples.

In the initial stages of the competition, we decided to use only the training dataset to build upon

our approaches. We further split the training set into train, dev, and test sets for evaluation purposes using an 80:10:10 split (Div A). Once we tested and finalized our approaches, we combined the train and test set of Div A with the trial set as our final training set (Div B). Due to the small size of the dataset, these additions to the training set of Div A will positively impact the model performance. However, to ensure that we could compare our final models with our previous results, we transfer the dev set directly to Div B. Further details regarding the constitution of these splits is provided in the Appendix A.

### 3.2 Pre-processing

**Tokenization:** For the sake of preserving the token spans, we first tokenized our data and then performed data cleaning. For tokenizing, we used the NLTK TreebankWord Tokenizer<sup>2</sup>, which is a rule-based tokenizer that tokenizes text on spaces and punctuation, hence preserving the original form of the words.

**Data Cleaning:** We then cleaned each token using different operations such as expanding contractions and removing digits and full stops.

## 4 Proposed Approach

### 4.1 Methodology

Pre-trained transformer models built using the transformer architecture (Vaswani et al., 2017) have been able to achieve, via transfer learning techniques, SOTA performance for most NLP tasks in recent times. We fine-tuned pre-trained transformer models with linear classifier head for performing sequence labeling for this task, which meant performing subtoken-level classification (Fig. 1a). Our baseline model used the pre-trained BERT-Base-Cased model, fine-tuned with cross-entropy loss and AdamW optimizer. The different hyperparameter values used for training the baseline and all subsequent models are reported in the Appendix B to facilitate replication of results. Subsequently, we improved upon this baseline using two techniques, semi-supervised learning and Self Adjusting Dice Loss. Along with this, we fine-tuned multiple different transformer models like BERT (Devlin et al., 2019), Electra (Clark et al., 2020), DistilBERT (Sanh et al., 2020), and XLNet (Yang et al.,

<sup>2</sup>[https://www.nltk.org/\\_modules/nltk/tokenize/treebank.html](https://www.nltk.org/_modules/nltk/tokenize/treebank.html)



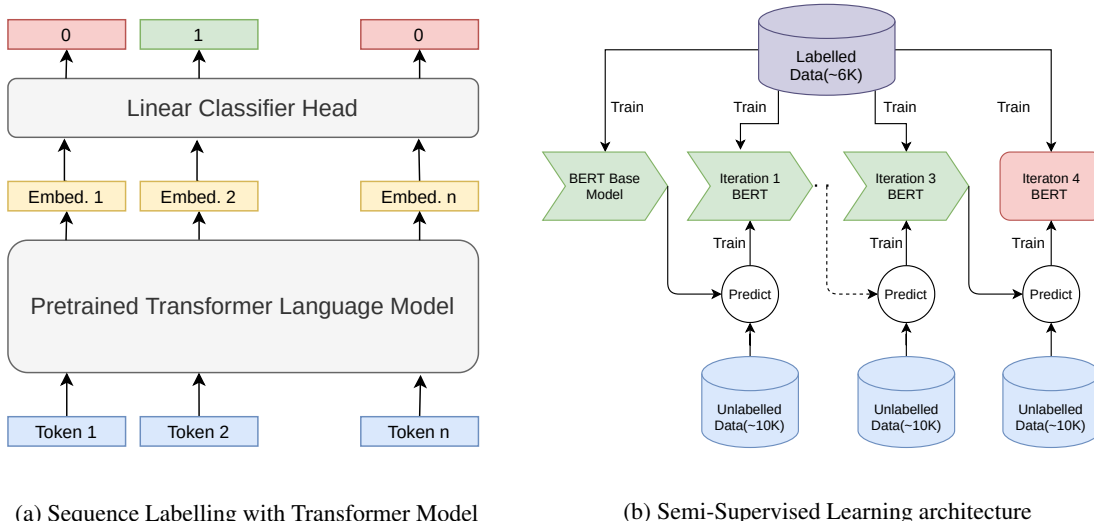


Figure 1: Model architectures

2020) for our Dice-Loss Approach and found differences in the predictions of the different transformer models to be beneficial for the final ensemble.

**Self Adjusting Dice Loss** One of the main issues with our dataset was that of class imbalance. For the sub-tokens derived from the BERT-Base-Cased tokenizer, the ratio of toxic to non-toxic sub-tokens was 1:10.16. However, we could not tackle this issue with over/under-sampling due to the nature of our problem, and training with a weighted cross-entropy loss function did not improve results. Therefore, we experimented with training with the Self-Adjusting Dice Loss (Li et al., 2020) which was proposed as an objective function for dealing with imbalanced datasets in NLP. The original dice coefficient is an F1-oriented statistic used to gauge the similarity of two sets. The paper proposed a loss function based on a modified dice coefficient, which they reported to achieve a better F1 score than models trained with cross-entropy loss.

$$DL = 1 - \frac{2(1 - p_{i1})^\alpha (p_{i1}) \cdot y_{i1} + \gamma}{(1 - p_{i1})^\alpha (p_{i1}) + y_{i1} + \gamma}$$

Here, for the  $i_{th}$  training instance,  $p_{i1}$  is the predicted probability of positive class and  $y_{i1}$  is the ground truth label. The loss function also has two hyperparameters, alpha and gamma, which we tuned for our models.

**Semi-Supervised Learning** The Civil Comments Dataset from which our training data was extracted consists of over 1 million comments; however, due to annotation constraints, the training set only had 7000 data samples. (Shams, 2014)

have shown that for text classification tasks, unlabelled data from a suitable data source could be used to train semi-supervised models that achieve better results than a model trained using supervised learning. Also, (Jurkiewicz et al., 2020) showed that the semi-supervised learning technique of self-training could improve performance on span identification tasks. Hence, we extracted 40000 toxic samples from the Civil Comments Dataset, which were labeled with a toxicity score of 0.7 or higher, and used these to perform four iterations of semi-supervised model training (Fig. 1b). We exhaustively divided the unlabelled samples into four batches of 10000 each and used each batch for exactly one iteration. As shown in Fig. 1b, for each iteration, pseudo labels were predicted for the complete batch using the model trained in the previous iteration, then these pseudo-labels along with the ground truth training labels were used to train the next model. For this approach, we only fine-tuned one transformer model, namely the pre-trained BERT-Base-Cased model.

## 4.2 Post-preprocessing

After obtaining the sub-token level labels from our model, we post-processed the results to convert them into an array of toxic character offsets. To perform this, we had mapped each sub-token to its offset span during tokenization and used that to retrieve the offsets of all the characters in the toxic sub-tokens. We also include all characters lying between two consecutive sub-tokens if both the sub-tokens are marked toxic. This was necessary

as spaces and punctuation were included in the toxic spans given by the annotators, as shown by the results in section 6.

## 5 Experiments and Results

The competition used the span level F1 score, calculated individually for each text sample from its character offsets and averaged over all the text samples, as the metric to evaluate system performance. We decided to use this metric for hyperparameter tuning and reporting the final results. However, during the training process, the models were checked for overfitting using the token level F1 score, which was also a good indicator of model performance as our training approach was that of a sequence labeling task.

The first set of experiments we performed were on the Div-A dataset. Our baseline model achieved an F1 score of 0.669 on the dev split on this set. The organizers also released a baseline model, consisting of a Spacy statistical model trained on the competition training dataset and evaluated on the competition trial dataset. The organizer’s baseline achieved an F1 score of 0.600 on the trial dataset. This score was significantly lower than that of our baseline model, and therefore we use our baseline model only to compare the performance of our subsequent models.

We then fine-tuned a BERT-Base-Cased model with the Self-Adjusting Dice Loss and AdamW optimizer and tuned the loss function’s two hyperparameters. The scores we obtained for the different hyperparameter values are reported in Table 9 in Appendix C. We got our best performing model with the hyperparameter values alpha-0.7 and gamma-0.25, achieving an F1 score of 0.6725 on the dev split.

Model	Dev F1 Score
BERT-Base-Cased	0.669
SSL Iteration-1	0.6837
SSL Iteration-2	0.6842
SSL Iteration-3	0.6882
SSI Iteration-4	<b>0.6893</b>

Table 1: Results for Semi-Supervised learning model

Next, we trained the BERT-Base-Cased model on the semi-supervised learning paradigm with cross-entropy loss and AdamW optimizer. For the first iteration, we used our baseline model to compute the pseudo labels. The model achieved

improved results with each iteration (Table 1), and our final model was scoring 0.6893 on the dev split.

To end this stage of experimentation, we computed the results on the test split of the Div-A dataset. We were able to make two inferences. Firstly the semi-supervised learning model had the best performance with an F1 score of 0.6774 on the test set but had a significantly worse score than it had on the dev set. Secondly, the dice loss trained model performed significantly better than the cross-entropy trained baseline with an F1 score of 0.662 compared to 0.648.

After this, we changed to the Div-B dataset and trained multiple different transformer models with the Self Adjusting Dice Loss. We found that the BERT-Base-Cased, Electra-Small, Electra-Base, and DistilBert-Base-Uncased models had peak performance for the hyperparameter values alpha-0.7 and gamma-0.25. However, for the XLNet-Base model, peak performance was achieved for alpha-0.4 and gamma-0.25. On further experimentation with these models, we also found that adding a full stop to the text samples during evaluation provided consistently better results on the dev set. The results obtained have been reported in Table 2.

Model	WFS	FS
BERT-Base-Cased	0.6754	0.6827
Electra-Small	0.6813	0.6861
Electra-Base	0.6776	0.6846
DistilBERT-Base-Unc.	0.6749	0.6773
XLNet-Base	0.6798	0.6852
SSI Iteration-4	0.6893	0.6932

Table 2: Effect of full stop on dev set during evaluation. Here WFS and FS represent without full stop and with full stop resp.

The final results of models trained either on modified Dice Loss or using Semi-Supervised learning, with full stop added during evaluation, are reported on the Div-B dev split and the competition test set in Table 3.

## 6 Ablation Study

After the competition, we wanted to study the effect of our different preprocessing and postprocessing techniques. We employ three main data cleaning techniques during our preprocessing, expanding contractions, removing numbers, and removing full stops. To study each particular technique’s impact, we created three new Div-B datasets, each having

	Model	Dev	Test
1	BERT-Base-Cased*	0.6827	0.668
2	Electra-Small*	0.6861	0.6771
3	Electra-Base*	0.6846	0.6720
4	DistilBERT-Base-Unc.*	0.6773	0.6822
5	XLNet-Base*	0.6852	0.6757
6	SSI Iteration-4	0.6932	0.672
	Ensemble (1,2,3,4,5,6)	0.6927	<b>0.6895</b>

Table 3: F1 Score on dev and competition test set  
\* - Models trained with modified Dice Loss

one of the preprocessing techniques missing. We then trained BERT-Base-Cased and Electra-Small models with Self Adjusting Dice Loss on each of these sets and evaluated the performance on their respective dev sets. The results are reported in Table 4 with the following acronyms:

- **TD** - All preprocessing steps used
- **WNUM** - Without removing numbers
- **WFS** - Without removing fullstops
- **WCON** - Without expanding contractions

Dataset	BERT-Base-Cased	Electra-Small
TD	0.6754	0.6813
WNUM	<b>0.6781</b>	0.6809
WFS	0.6713	0.6743
WCON	0.671	<b>0.6829</b>

Table 4: F1 Score for different preprocessing techniques on dev set

The results show that removing numbers and expanding contractions both had contrasting effects on the two models. This shows that we could have yielded better results by trying different preprocessing techniques for the different transformer models. Apart from that, we see that the most positive effect on model performance came from removing full stops from the training data in both cases.

We also wanted to see the effect of our post-processing step. For that, we compared the performance of the BERT-Base-Cased model on the Div-B dev split. As expected, the results showed minor improvement due to our postprocessing as the score increased from 0.6748 to 0.6754.

## 7 Error Analysis

The results we have obtained have brought to light some problems that need to be resolved. First of all, the data annotations have many issues, leading to a lower F1 score even though the predicted

Example Set	Val	Test
E.S Val:41 Test:394	0.0731	0.0380
N.E.S Val:753 Test:1606)	0.7265	0.8493
All Val:794 Test :2000	0.6927	0.6895

Table 5: System performance over empty span (E.S) and non-empty span(N.E.S) examples over Div-B split

toxic spans are more appropriate in many cases. We have included some examples in Appendix D. In some cases, the annotations are not uniform in what toxicity label they assign to the same word over different text samples. We have also observed that complete sentences were marked as toxic just because of the presence of a few toxic words in them. These irregularities in the annotations make it difficult for the model to generalize on the data.

Besides the incorrect annotations, we further try to analyze the type of mistakes our system is making. The dataset contains numerous examples where no toxic spans are annotated. Such a case arose when the annotators had difficulty in attributing toxicity to a particular span. Investigating our model performance shows that our model highly under-performs on such examples. Table 5 depicts the drastic difference in the performance of the system over empty span examples (E.S) and non-empty span examples (N.E.S). Upon closely following E.S examples, we discovered that annotations of such examples carry more subjectivity than the others. In such cases, our model usually labels the word with the most negative sentiment as toxic and thus performs poorly.

In addition to the empty span examples, we also discover that our model fails to capture the full context in some cases. For e.g., in the phrase “no more Chinese,” our model only predicts the word Chinese as toxic, whereas the complete phrase attributes to the toxicity of the sentence. Another problem is our model’s inconsistency in labeling the corresponding noun and adjective pairs in a sentence. However, similar types of inconsistencies were also found in the annotations and are therefore difficult to avoid [Appendix D].

## 8 Conclusion

The task of detecting toxic spans in the text is a novel one, and there is no doubt about how impor-

tant a model trained successfully for this task can turn out to be for online content moderation. However, the data gathered from online platforms tend to be noisy and corrupted. Coupled with the limitations of generating large-scale annotated datasets in real life, they pose two daunting challenges. In conclusion, our final submission shows that transfer learning through pre-trained transformer models can achieve competitive results for this task. Using modified loss functions and semi-supervised learning, even more can be extracted from limited annotated data. Moreover, considering the subjectivity involved in span detection, the task can also be expanded to report severity scores of spans and classify the type of toxicity. This will further help simplify and rationalize online content moderation.

## References

- Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval 2017 task 10: Scienceie-extracting keyphrases and relations from scientific publications. *arXiv preprint arXiv:1704.02853*.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Electra: Pre-training text encoders as discriminators rather than generators](#).
- Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*.
- Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.
- Karishma Laud, Jagriti Singh, Randeep Kumar Sahu, and Ashutosh Modi. 2020. problemconquero at semeval-2020 task 12: Transformer and soft label-based approaches. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2123–2132.
- Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. [Dice loss for data-imbalanced nlp tasks](#).
- Ping Liu, Wen Li, and Liang Zou. 2019. Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91.
- Xiaohua Liu, Shaodian Zhang, Furu Wei, and Ming Zhou. 2011. [Recognizing named entities in tweets](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 359–367, Portland, Oregon, USA. Association for Computational Linguistics.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.
- David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.
- Sean Papay, Roman Klinger, and Sebastian Padó. 2020. Dissecting span identification tasks with performance prediction. *arXiv preprint arXiv:2010.02587*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.
- Georgios K Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. Detecting offensive language in tweets using deep learning. *arXiv preprint arXiv:1801.04433*.
- Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.
- Mujahed A Saif, Alexander N Medvedev, Maxim A Medvedev, and Todorka Atanasova. 2018. Classification of online toxic comments using the logistic regression and neural networks models. In *AIP*

- conference proceedings*, volume 2048, page 060011. AIP Publishing LLC.
- Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).
- Alessandro Seganti, Helena Sobol, Iryna Orlova, Hannam Kim, Jakub Staniszewski, Tymoteusz Krumholz, and Krystian Koziel. 2019. [NLPR@SRPOL at SemEval-2019 task 6 and task 5: Linguistically enhanced deep learning offensive sentence classifier](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 712–721, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Rushdi Shams. 2014. [Semi-supervised classification for natural language processing](#). *CoRR*, abs/1409.7612.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#).
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. [Xlnet: Generalized autoregressive pretraining for language understanding](#).
- David Yarowsky. 1995. [Unsupervised word sense disambiguation rivaling supervised methods](#). In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [Semeval-2019 task 6: Identifying and categorizing offensive language in social media \(offenseval\)](#). *arXiv preprint arXiv:1903.08983*.

## Appendix

### A Dataset

We worked on two different splits of the data across different stages of competition. Table 6 represent the no. of examples in train, val and test across Div-A and Div-B split.

	Div-A	Div-B
Train	6351	7835
Dev	794	794
Test	794	2000
Total	7939	10629

Table 6: Distribution of examples across Div-A and Div-B split

Div-A is basically a 80:10:10 split of the training data released by the organisers whereas Div-B split uses the train and test set of Div-A along with competition trial data as its training set. Div-B uses the official test set as its test set while keeping the dev set same as that of Div-A.

### B Model Training

In this section, we provide the hyperparameter values we used while training our final models to facilitate the replication of our results at a later time. The acronyms correspond to:

- **LR** : Learning Rate
- **ML** : Max Len
- **LC** : Data Lowercase
- **DL** : Dice Loss (Alpha, Gamma)

Hyperpara.	BERT-Base-Cased	Electra
LR	1E-5	3E-5
ML	500	500
LC	False	True
DL	0.7,0.25	0.7,0.25

Table 7: Hyperparameter Values for BERT-Base-Cased and Electra (Small and Base)

For baseline model and semi-supervised learning model, the cross-entropy loss function provided in PyTorch was used with default hyperparameters. For AdamW optimizer, we used weight decay rate of 0.01 for all model parameters apart from the bias, gamma and beta terms, for which we used 0 weight decay.

Hyperpara.	Distil-Base-Unc.	XLNet-Base
LR	1E-5	3E-5
ML	500	400
LC	True	False
DL	0.7,0.25	0.4,0.25

Table 8: Hyperparameter Values for Distil-Base-Unc. and XLNet-Base

## C Results

### C.1 Dice Loss Results

In this section, we report the results of our hyperparameter tuning for training with Self Adjusting Dice Loss. The result reported are for BERT-Base-Cased model on the Div-A dataset.

Parameter Values	Dev F1 Score
Alpha-0, Gamma-1	0.665
Alpha-0, Gamma-2	0.657
Alpha-0.5 Gamma-1	0.652
Alpha-0.7, Gamma-1	0.671
Alpha-0.8, Gamma-1	0.668
Alpha-0.7, Gamma-0.5	0.6723
Alpha-0.7, Gamma-0.25	<b>0.6725</b>

Table 9: BERT-Base-Cased trained with Self Adjusting Dice Loss on Div-A train split

## D Error Analysis

We manually crawled through a part of dataset and found several inadequate annotations as described through table 11. Apart from erroneous annotations, we also found some examples of inconsistency in the structure of annotations. Table 10 outlines a few such pairs where different strategies are used for annotating similar texts. These examples highlight the inconsistencies in labelling of corresponding noun-adjective pairs.

Text	Ground Truth
You are an idiot	['idiot']
You are truly witless	All tokens
F***ing moron ....	['F***ing']
Fucking moron needs...	['Fucking', 'moron']
Hilary is a liar...	All tokens
Trump is an idiot.	['idiot']

Table 10: Irregularity in annotation for similar text-samples.

Text	Ground Truth	Suggested
I have no doubt it was Photoshopped(by an idiot, for an idiot)	['idiot']	['idiot', 'idiot']
She's a Marxist twit.	['Marxist']	['Marxist', 'twit' ]
Move to Russia and be a fascist pig all you want.	All tokens	['fascist', 'pig']
lame. really really lame and cheap shot. pathetic actually.	['lame', 'lame', 'actually']	['lame', 'lame', 'pathetic']
So "f*** you all", "kiss my a—" and ...	['f***']	['f***', 'a-']
You are just a stick buthole!	[]	['stick', 'buthole']

Table 11: Shows the words labelled toxic as ground truth(from annotators) and suggested accurate annotations

# UniParma at SemEval-2021 Task 5: Toxic Spans Detection Using CharacterBERT and Bag-of-Words Model

Akbar Karimi

Leonardo Rossi

Andrea Prati

University of Parma

{akbar.karimi, leonardo.rossi, andrea.prati}@unipr.it

## Abstract

With the ever-increasing availability of digital information, toxic content is also on the rise. Therefore, the detection of this type of language is of paramount importance. We tackle this problem utilizing a combination of a state-of-the-art pre-trained language model (CharacterBERT) and a traditional bag-of-words technique. Since the content is full of toxic words that have not been written according to their dictionary spelling, attendance to individual characters is crucial. Therefore, we use CharacterBERT to extract features based on the word characters. It consists of a Character-CNN module that learns character embeddings from the context. These are, then, fed into the well-known BERT architecture. The bag-of-words method, on the other hand, further improves upon that by making sure that some frequently used toxic words get labeled accordingly. With a  $\sim 4$  percent difference from the first team, our system ranked 36<sup>th</sup> in the competition. The code is available for further research and reproduction of the results<sup>1</sup>.

## 1 Introduction

The user generated digital content is increasing rapidly every second of the day. This can include some toxic language whose detection can be difficult due to the complexities of human languages. We address this problem by participating in SemEval Workshop 2021 Task 5 (Pavlopoulos et al., 2021).

In many cases, the data, which are considered to be toxic, contain words that have not been written in their standard forms. There might also be a lot of misspelling or letter replacements. In addition, usually the words that are considered to be the most offensive are bleeped which makes them difficult to be recognized if we use a model which learns the

content representation based on the words. Apart from word related issues, the context also plays a crucial role in the meaning that a word conveys since words in different contexts can have various meanings.

Therefore, in order to cope with these issues, we opt for a recently pre-trained language model which has been trained on character level. CharacterBERT (El Boukkouri et al., 2020) is a deep neural network model that has been pre-trained on Wikipedia and OpenWebText (Gokaslan and Cohen) corpora using the BERT architecture (Devlin et al., 2019) with an addition of a character-aware Convolutional Neural Network (CNN) (Kim et al., 2016; Peters et al., 2018). BERT-based models have now become pervasive in many different natural language processing tasks such as reading comprehension (Xu et al., 2019), named entity recognition (Liang et al., 2020), sentiment analysis (Karimi et al., 2020), and language understanding (Jiao et al., 2020) as well as similar ones to toxic language detection such as propaganda detection (Jurkiewicz et al., 2020). While the BERT model is beneficial in extracting the contextual information from the text on the word level, the character-aware CNN attends to the individual letters which helps in dealing with out-of-vocabulary, unknown and rare words.

In addition to using a deep language model for detecting toxic language, we employ a very simple Bag-of-Words model that can achieve a close performance to that of the deep model. By building a dictionary of toxic words from the training data and by taking into account their frequency and ratio of toxicity, we come up with a simple model that performs as closely as about 2 percent difference in performance to the deep model’s result. Moreover, we improve the results of CharacterBERT by combining it with the output of a version of the Bag-of-Words model.

<sup>1</sup><https://github.com/IMPLabUniPr/UniParma-at-semeval-2021-task-5>



spans	text
[8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39]	Another violent and aggressive immigrant killing a innocent and intelligent US Citizen.... Sarcasm
[0, 1, 2, 3]	Damn, a whole family. Sad indeed.
[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]	What a knucklehead. How can anyone not know this would be offensive??

Table 1: Three examples from the training set

## 2 System Description

Our system consists of four main stages namely, pre-processing, applying CharacterBERT, applying Bag-of-Words model, and finally combining the results of the two models. We describe each of these stages in the following subsections.

### 2.1 Pre-processing

The training dataset consists of rows of various lengths and an array of character spans indicating their toxic parts. Each row can contain several sentences. Table 1 shows three examples of the training data.

We approach the task of toxic spans detection as a sequence labelling task where each word of the input row is classified into one of the predefined classes. We define three classes of {B, I, O}, meaning that each word can be the first word (B) of a set of continuous toxic words, in between (I), or not toxic (O). Therefore, in order for both of our models to be able to process these inputs, we first need to break the rows into words and label them as one of the above-mentioned classes. This was carried out by simply splitting each input row at the *space* characters. Then, after creating a dataset that has been labeled on the word level, we can use it as the input of our models. The same is done for the Bag-of-Words model with a difference in treating the bleeped words which is described in Subsection 2.3.

### 2.2 CharacterBERT

CharacterBERT model is almost identical to the well-known BERT model with a difference in initial embedding. In the general BERT model, words are broken into pieces and the embeddings for these word pieces are computed. In CharacterBERT, however, words are divided into letters or characters. Then, using CNN modules the embeddings are computed on the character level (Figure 1). This makes the network extract features on the

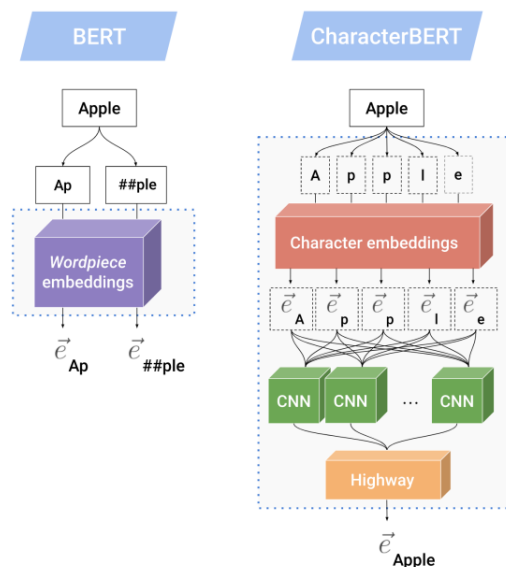


Figure 1: The difference between the BERT and CharacterBERT models is the way they compute the initial embeddings. The former uses word-piece embeddings while the latter uses character embeddings. Figure taken from El Boukkouri et al. (2020).

lowest level, making it suitable for contexts which contain many unseen vocabulary terms such as misspelled words or technical jargon. After the initial character-aware CNN layer, there is the BERT<sub>base</sub> architecture which contains 12 layers (blocks) of Transformer (Vaswani et al., 2017) with the hidden size of 768 and 12 attention heads. The final layer representations are converted into logits using a fully connected layer after which a Softmax layer is applied to extract the token’s (word’s) class.

### 2.3 Bag-of-Words Model

This model is a simple script of fewer than 80 lines of code. However, its performance on the Toxic Spans Detection task can get very close to the CharacterBERT model which has millions of parameters. In this model, by examining the training set, we first build a dictionary of toxic words with their frequency. Table 2 presents the top ten words of

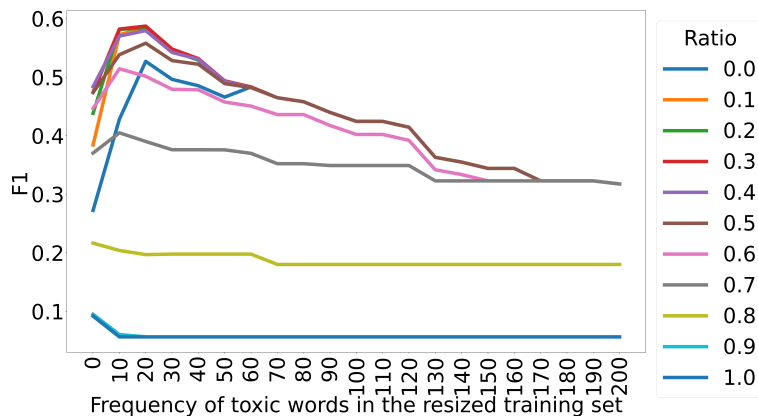


Figure 2: Performance of the Bag-of-Words model on validation set. The frequencies and ratios are the minimum thresholds that specify whether or not to consider a word toxic.

Word	Frequency
stupid	973
idiot	557
idiots	353
stupidity	223
ignorant	190
dumb	157
moron	147
fool	141
pathetic	138
crap	121

Table 2: Top 10 toxic words in the training set

this dictionary in terms of frequency.

Then, we locate the words from the toxic dictionary in each sentence of the test set. If the word is found and its frequency as well as its toxicity ratio in the training set are higher than certain values, it is labeled as toxic. This ratio which we call *toxicity ratio* (defined below) along with the *term frequency* are the only parameters of the Bag-of-Words model.

$$\text{toxicity ratio} = \frac{\text{labeled as toxic frequency}}{\text{total frequency}}$$

The test dataset also contains words that are bleeped. Since these words can be considered toxic with a high certainty (otherwise they would not be bleeped), we extract them separately from the test set and label them directly as toxic.

## 2.4 Combining the Two Models

In order to get the improved version of the toxic language labeling, the union of the spans detected by

the bag-of-words model and that of CharacterBERT is taken. The results will improve if there are words labeled correctly with the Bag-of-Words model that are not in the output for CharacterBERT. This can be achieved by specifying a high toxicity ratio for a word to be labeled as toxic. Also, the wrongly labeled tokens should not be too many since it can have a negative effect on the F1 score. Therefore, the frequency with which a toxic word appears should be somewhat high. Striking a balance between these two parameters can help improve the output of CharacterBERT.

## 3 Experiments and Results

### 3.1 Performance of CharacterBERT

We ran the experiments for the general domain CharacterBERT with its default setting on a GPU (GeForce RTX 2070) which had 8GB of memory. We specified batch sizes of 4 for both training and testing and fine-tuned it on the toxic data only for one epoch which produces an F1 score of 65.13. It is worth noting that more training did not improve the performance.

### 3.2 Analysis of the Bag-of-Words model

In order to experiment with the Bag-of-Words model, we divide the original training set into a resized training set with 7000 sentences and a validation set with 939 sentences which were taken from the end of the original training set. Then, we find the best parameters on the validation set and using those parameters on the test data, we get a performance of almost 63 percent which is only 2 percent smaller than our deep model. Figure 2

Model	F1
CharacterBERT	65.13
BoW (v1)	51.75
BoW with best parameters (v2)	62.79
CharacterBERT + BoW (v2)	65.87
<b>CharacterBERT + BoW (v1)</b>	<b>66.72</b>

Table 3: Comparing results of the proposed models. The boldfaced one is the submitted version. BoW: Bag-of-Words model.

Word	Frequency	Toxicity Ratio
stupid	973	0.78
idiot	557	0.84
idiots	353	0.81
stupidity	223	0.77
moron	147	0.71
idiotic	98	0.74
hypocrite	75	0.88
shit	56	0.72
scum	52	0.70
hypocrites	44	0.76

Table 4: Words selected as the toxic words with minimum frequency of 40 and minimum toxicity ratio of 0.7 (BoW (v1))

shows this model’s performance for its two parameters on the validation set. One parameter represents the *minimum* frequency with which a toxic word appears in the resized training set and the other one is its *minimum* toxicity ratio in the resized training data.

We can see from Figure 2 that the best results are achieved when the minimum frequency is 20 and the minimum ratio is 0.3 or 0.4. Since a larger ratio can be a sign of more toxicity, we choose 0.4 as the ratio and a frequency of 20 as the thresholds with which we apply the model on the test set. This gives an F1 score of 62.79 percent (Table 3) which is not that much below the result of the deep model.

We can also see from Table 3 that although combining the output of the Bag-of-Words model with that of CharacterBERT improves the results a little bit, it is still is not as significant as the first version. In the first version of the Bag-of-Words model, which was found during our primary experiments, the minimum word frequency is 40 and the minimum toxicity ratio is 0.7. With these parameters, only 10 words are selected from the training set. The frequency and toxicity ratio of these words can be seen in Table 4.

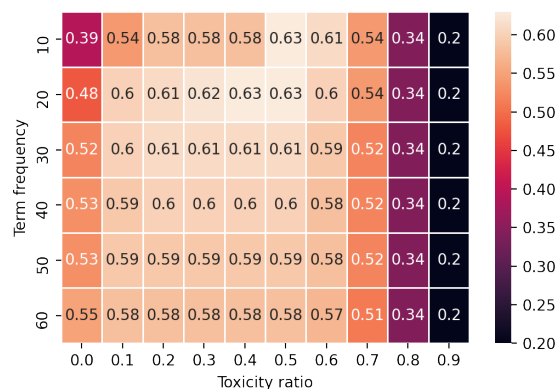


Figure 3: Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio before combining with CharacterBERT

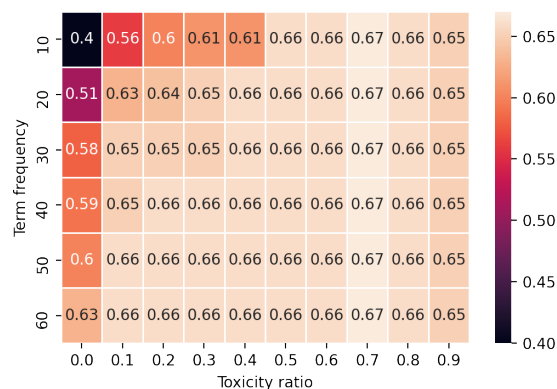


Figure 4: Heatmap of the results (F1 scores) with different values of term frequency and toxicity ratio after combining with CharacterBERT

Although the performance of this version is a lot lower than the second version (v2) of the BoW model, it helps to improve the performance of CharacterBERT. The reason for this behavior can be attributed to the fact that models with higher thresholds both in terms of frequency and toxicity ratio tend to output more certain results, albeit fewer words than the ones that should be labeled as toxic. Therefore, many toxic words that are less probable are not extracted and F1 score drops.

Looking at Figure 3, we can see that, indeed, the best parameters from the experiments on the validation set (ratios 0.3 and 0.4 with frequencies 10 and 20) yield some of the best results on the test set. However, when these results are combined with the output of the CharacterBERT, we see that the higher the toxicity ratio the better the results (Figure 4) until 0.7 which gives the maximum im-

provement. The 0.8 ratio makes the predictions still a little better but 0.9 does not affect them since the words that are labeled as toxic with this certainty have most probably been found also by CharacterBERT.

## 4 Conclusion

We described the system we utilized to detect toxic language. In our approach, we first fine-tune CharacterBERT, a character-level pre-trained language model, on the toxic training data. Then using a simple bag-of-words model, we further improve the results of this system. The Bag-of-Words model labels the words based on their frequency and the ratio of toxicity in the training data. We showed that this model, although extremely simple, gives a close performance to that of CharacterBERT with millions of parameters.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun’ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915.
- Aaron Gokaslan and Vanya Cohen. Openwebtext corpus.
- Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Gralinski. 2020. ApplicaAI at semeval-2020 task 11: On RoBERTa-CRF, span CLS and whether self-training helps them. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1415–1424.
- Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2020. Improving BERT performance for aspect-based sentiment analysis. *arXiv preprint arXiv:2010.11731*.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. 2016. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.
- Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Hu Xu, Bing Liu, Lei Shu, and S Yu Philip. 2019. BERT post-training for review reading comprehension and aspect-based sentiment analysis. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2324–2335.

# UPB at SemEval-2021 Task 5: Virtual Adversarial Training for Toxic Spans Detection

Andrei Paraschiv, Dumitru-Clementin Cercel, Mihai Dascalu

University Politehnica of Bucharest, Faculty of Automatic Control and Computers

andrei.paraschiv74@stud.acs.upb.ro  
{dumitru.cercel, mihai.dascalu}@upb.ro

## Abstract

The real-world impact of polarization and toxicity in the online sphere marked the end of 2020 and the beginning of this year in a negative way. Semeval-2021, Task 5 - Toxic Spans Detection is based on a novel annotation of a subset of the Jigsaw Unintended Bias dataset and is the first language toxicity detection task dedicated to identifying the toxicity-level spans. For this task, participants had to automatically detect character spans in short comments that render the message as toxic. Our model considers applying Virtual Adversarial Training in a semi-supervised setting during the fine-tuning process of several Transformer-based models (i.e., BERT and RoBERTa), in combination with Conditional Random Fields. Our approach leads to performance improvements and more robust models, enabling us to achieve an F1-score of 65.73% in the official submission and an F1-score of 66.13% after further tuning during post-evaluation.

## 1 Introduction

Nowadays, online engagement in social activities is at its highest levels. The lockdowns during the 2020 COVID-19 pandemic increased the overall time spent online. In Germany for instance, Lemenager et al. (2021) observed that 71% of considered subjects increased their online media consumption during this period. Unfortunately, online toxicity is present in a large part of the social and news media platforms. As such, automated early detection is necessary since toxic behavior is often contagious and leads to a spillover effect (Kwon and Gruz, 2017).

Recently, a significant effort was put into the detection of toxic and offensive language (van Aken et al., 2018; Paraschiv and Cercel, 2019; Tanase et al., 2020b,a), but the challenging nature of these problems leaves several avenues unexplored. In addition, most shared tasks focus

on the distinction between toxic/non-toxic (Wulczyn et al., 2017; van Aken et al., 2018; Juuti et al., 2020) or offensive/non-offensive posts in various languages (Struß et al., 2019; Zampieri et al., 2019a,b, 2020; Mandl et al., 2020; Aragón et al., 2020). The Semeval-2021 Task 5, namely Toxic Spans Detection (Pavlopoulos et al., 2021), tackles the problem of identifying the exact portion of the document that gives it toxicity. The provided dataset is a subset of the Jigsaw Unintended Bias in Toxicity Classification dataset<sup>1</sup>, with annotated spans that represent toxicity from a document.

In this paper, we describe our participation in the aforementioned Toxic Spans Detection task using several Transformer-based models (Vaswani et al., 2017), including BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), with a Conditional Random Field (CRF) (Lafferty et al., 2001) layer on top to identify spans that include toxic language. We introduce Virtual Adversarial Training (VAT) (Miyato et al., 2015) in our training pipeline to increase the robustness of our models. Furthermore, we enhance part of our models with character embeddings based on the Jigsaw Unintended Bias dataset to improve their performance. Finally, we compare the proposed models and analyze the impact of various hyperparameters on their performance.

The rest of the paper is structured as follows. The next section introduces a review of methods related to toxic language detection, sequence labeling, and adversarial training (Kurakin et al., 2016). The third section discusses the employed models, as well as the VAT procedure. Results are presented in the fourth section, followed by discussions, conclusions, and an outline of possible future works.

<sup>1</sup><https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification>

## 2 Related Work

**Toxic Language Detection.** There are several research efforts to detect toxic texts based on the Jigsaw Unintended Bias dataset, out of which most focus on the Kaggle competition task - predicting the toxicity score for a document. [Morzhov \(2020\)](#) compared models based on Convolutional Neural Networks (CNNs) ([Kim, 2014](#)) and Recurrent Neural Networks ([Cho et al., 2014](#)) with a Bidirectional Encoder Representations from Transformers (BERT) architecture ([Devlin et al., 2019](#)), obtaining the best performance from an ensemble of all used models. [Gencoglu \(2020\)](#) and [Richard and Marc-André \(2020\)](#) used the same dataset to improve on the automatic detection of cyberbullying content.

**Sequence Labeling.** Predicting the type for each token from a document rather than providing a label for the whole sequence is a task often associated with named entity recognition ([Ma and Hovy, 2016](#)), but can be performed in other Natural Language Processing pipelines, including part-of-speech tagging ([Ling et al., 2015](#)) and chunking ([Hashimoto et al., 2017](#)). A common practice in sequence tagging models ([Peters et al., 2018](#); [Avram et al., 2020](#); [Ionescu et al., 2020](#)) is to use a CRF as a final decoding layer.

**Adversarial Training.** Researched first in image classification ([Szegedy et al., 2013](#)), adversarial examples are small input perturbations that are hardly distinguishable for humans, but can dramatically shift the output of a neural network. These examples can be used in adversarial training (AT) ([Goodfellow et al., 2014](#)) as a regularization method that can increase the robustness of the model. Using the worst-case outcome from a distribution of small norm perturbations around an existing training sample, a new data point is created and inserted into the training process.

Extending AT to a semi-supervised setting, VAT ([Miyato et al., 2016](#)) does not require label information for the adversarial examples. VAT aims to increase the local distributional smoothness by adding perturbations to the embedding output. Recently, several studies ([Kumar and Singh, 2020](#); [Liu et al., 2020](#); [Si et al., 2020](#)) focused on applying VAT in Transformer-based models and obtained improvements in comparison to baseline methods on several classification tasks.

## 3 Method

### 3.1 Corpus

The dataset for the competition is a subset of the Jigsaw Unintended Bias in Toxicity Classification English language corpus, with annotated spans that make the utterance toxic. From the 8,597 trial and train records, 8,101 had at least one toxic span. By cross-referencing with the original Jigsaw dataset which contains additional information, we retrieved the toxicity scores for each text and determined that the mean toxicity score for the train and test set were very close (0.8429 versus 0.8440; see [Figure 1](#) for corresponding kernel density estimates). Moreover, only 17 out of 2,000 test data rows had a toxicity score below 0.75. Nevertheless, an off-balance was noticed between the test and train set - 80.3% entries from the test set had at least one toxic span versus a considerably higher density of 94.2% in the train set.

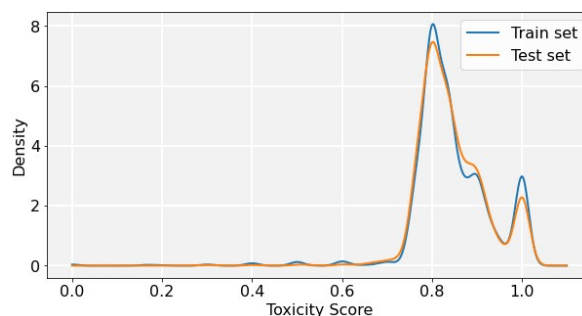


Figure 1: Kernel density estimate using Gaussian kernels for the toxicity scores in both the training and test data.

The training dataset was split into sentences while ensuring that there are no splits inside a toxic span and there are no sentences shorter than three words. Under these settings, our training dataset consists of a total of 26,589 sentences, including 10,117 records that contained toxic spans; 15% were selected for validation. Another 2,000 entries were provided by the competition organizers for testing; the labels for this dataset were made available after the competition.

For our unsupervised training samples, we selected 20,000 random records from the Jigsaw dataset, making sure there was no overlap with the Semeval-2021 training data. Additionally, we replaced all URL-s with a special token and applied lower case on all records.

Model	F1-score validation set	F1-score competition test set
LSTM-CRF-VAT	75.82%	62.49%
LSTM-CRF-VAT+chars	76.27%	63.65%
BERT-base-CRF	79.25%	62.32%
BERT-base-CRF-VAT	80.66%	64.59%
BERT-toxic-CRF-VAT*	<b>81.08%</b>	65.73%
BERT-news-CRF-VAT	80.80%	64.57%
BERT-news-CRF-VAT ( $\gamma=0.6$ )	81.01%	<b>66.13%</b>
BERT-news-CRF-VAT+chars	80.79%	64.57%
RoBERTa-large-CRF-VAT	78.13%	62.73%

Table 1: F1-scores for predictions on the validation and test set.

\* marks the model from the official submission.

### 3.2 Virtual Adversarial Training

The robustness of the model in Adversarial Training is improved through examples that are close to available training data, but the model would be likely to assign a different label than the training one, thus leading to loss increase. In VAT, Miyato et al. (2018) adapted the adversarial training from supervised to semi-supervised settings by adding an additional loss using the Kullback–Leibler divergence between the predictions of the original data and the same data with random perturbations. Since the output distributions are compared, the information about labels is not needed for the adversarial loss:

$$L_{adv} = KL(P(\hat{y}|e, \Theta) || P(\hat{y}|e + d, \Theta)) \quad (1)$$

where  $e$  is the embedding associated with the sample,  $d$  the perturbation, and  $\hat{y}$  is the predicted output.

True labels are required in general to compare the losses and find the worst case perturbations. However, this can be avoided by bounding the norm of the perturbation  $\delta$  to  $\eta$ ; thus, the value of the perturbation becomes:

$$d = \arg \max_{\delta; \|\delta\|_2 < \eta} KL(P(\hat{y}|e, \Theta) || P(\hat{y}|e + \delta, \Theta)) \quad (2)$$

Afterwards, we can estimate the perturbation  $d$  using also the gradient  $g$  and a hyperparameter  $\epsilon$  for the magnitude by applying the second-order Taylor approximation and a single iteration of the power method:

$$d = \frac{g}{\|g\|_2} \epsilon \quad (3)$$

where

$$g = \nabla_{\delta} KL(P(\hat{y}|e, \Theta) || P(\hat{y}|e + \delta, \Theta)) \quad (4)$$

In order to reduce the complexity and computation for the gradient, we ignore the dependency on  $\Theta$ . Also, the number of power iterations can be another hyperparameter for the model. The final loss function used by all models is a combination of the supervised and unsupervised adversarial loss:

$$L_{total} = \gamma L_{sup} + (1 - \gamma) L_{adv} \quad (5)$$

where  $\gamma$  is another tunable hyperparameter.

### 3.3 Implementation Details

In our experiments, pre-trained Transformer models are followed by a linear transformation of their last hidden state, and a final CRF layer. More precisely, we compare the effectiveness of several flavors of BERT models, alongside the VAT technique as follows: BERT base, a 768-dimensional model provided by Google (*BERT-base-CRF-VAT*), Unitary’s toxic BERT (Hanu and Unitary team, 2020) (*BERT-toxic-VAT*), BERT pre-trained on fake and hyperpartisan news (Paraschiv et al., 2020) (*BERT-news-CRF-VAT*), and *RoBERTa-large-CRF-VAT*, the equivalent of BERT-base-CRF-VAT that relies on RoBERTa instead of BERT.

In addition to these models, we experimented with enhancing the BERT-based representation with character embeddings (Kim et al., 2016). These character representations were trained on the entire Jigsaw dataset using a CNN-BiLSTM model (Ma and Hovy, 2016) with the next character prediction objective. We concatenated the

obtained character-level embeddings with the aforementioned Transformer’s last hidden state, and refer to this variant as *BERT-news-CRF-VAT+chars*.

As baseline systems, we design two methods: *LSTM-CRF-VAT* with GloVe embeddings (Pennington et al., 2014) and a *LSTM-CRF-VAT+chars* having character-level embeddings and VAT. In all BERT-based models, we used a maximum sequence length of 96 tokens and a sequence of 64 tokens for the LSTM baseline. Since the input words can consist of more than one token, we assign the toxicity label to a word if at least one component token is inferred as toxic.

The best hyperparameters for the BERT-base model were determined through grid search on the development set. The identified optimal values ( $\epsilon = 2$ ,  $\eta = 0.1$ , and two power iterations) were used in all other flavors;  $\gamma$  was set to 0.5 in the final loss function to balance both approaches. Furthermore, all BERT-based models were trained for one epoch in contrast with the LSTM-CRF-VAT and LSTM-CRF-VAT baselines that were trained for three epochs and four epochs, respectively.

## 4 Results

The evaluation metric for the Toxic Spans Detection task was an adapted version of the F1-score (Da San Martino et al., 2019) that takes into account the size of the overlap between prediction spans and golden labels.

Results for all developed models with the aforementioned hyperparameters (i.e.,  $\gamma = 0.5$ ,  $\epsilon = 2$ ,  $\eta = 0.1$ , and two power iterations) are presented in Table 1. Since the training data had a slightly different distribution of the span density, part of our models that performed worse on our dev set performed better on the competition test set. Adding the character embedding representation to BERT-based models did not prove to be of use in our pre-evaluation tests, but in post-evaluation, we noticed that slightly tweaking the  $\gamma$  hyperparameter for the loss from 0.5 to 0.6 brought the F1-score to 66.13%. Despite performance on the validation set was insensitive to the change in  $\gamma$  between 0.5 and 0.6, the results on the test set were more than 1.5% apart. This is mostly due to the unsupervised training that is strengthening the model’s confidence on edge cases which would lower its precision.

Figure 2 introduces the influence of the perturbation magnitude  $\epsilon$  on the overall performance of three models. The impact of  $\epsilon$  in the adversarial

training effectiveness is significant, but it is also highly dependant on the used model and can only be determined experimentally.

Our models performed well on the detection task, learning not only common toxic expressions like "moron", "stupid", "pathetic troll", "disgusting", "hang-em high", but also obfuscated expressions like "f\*cking nasty" and "b\*tchy". Nonetheless, the models fail to detect more obscured words like "you don't know s\*\*\*" or "Kill this F'n W\*ore on site". All models have the tendency to over-predict toxicity by adding words to the toxic expression - for example, "What a pile of shit" was automatically labeled as "What a pile of shit".

The character-level embeddings boosted the performance of the baseline LSTM-CRF-VAT model but did not improve any BERT model since it leads to detecting longer spans as toxic (see Table 2) which in return lowers precision.

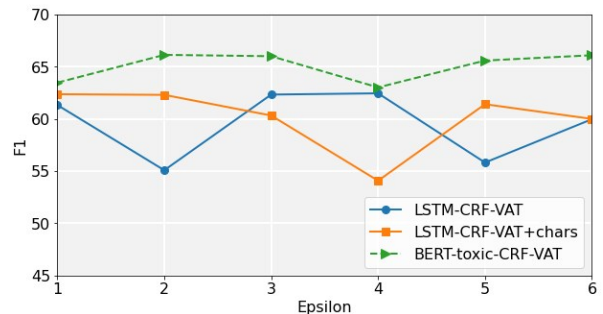


Figure 2: Impact of the perturbation magnitude  $\epsilon$  on the F1-scores for the predictions on the official test set.

## 5 Discussions and Error Analysis

In this section, we analyze the BERT-toxic-CRF-VAT performance versus the golden label values from the competition test set. The precision and recall for our best model are 65.74% and 85.54%, respectively, which are indicative of a tendency to over-predict toxic spans. As we mentioned in section 3.1, even though almost all documents in the test set had a high toxicity score in the original Jigsaw dataset, many had no annotated toxic spans. Indeed, there were 295 records where our model detected a toxic span and none were labeled in the test set. Table 3 includes examples of such detected spans those labeling is subjective and other detection errors. Words like "stupid", "dumb", and "crap" were assigned as toxic throughout the test data due to their high presence in the training data spans. There were also milder errors, spans that overlap with the golden labels, but the model omits



Annotated sample	BERT-news-CRF-VAT+chars	BERT-toxic-CRF-VAT
Greedy pig strikes again!	Greedy pig strikes again!	Greedy pig strikes again!
What's honorable is telling idiots that they're idiots (even if tardily)	What's honorable is telling idiots that they're idiots (even if tardily)	What's honorable is telling idiots that they're idiots (even if tardily)
A fool and our money are soon to be parted.	A fool and our money are soon to be parted.	A fool and our money are soon to be parted.

Table 2: Examples of model predictions with character embeddings (BERT-news-CRF-VAT+chars) and without (BERT-toxic-CRF-VAT), compared to the official annotations. The toxic spans are highlighted.

Annotated sample	Model predictions
As is Drumpf a lying gasbag.	As is Drumpf a lying gasbag .
The Liberals should listen to the economic experts and dramatically increase immigration levels (at least 500,000 per year). Stop listening to the loser racist trash.	The Liberals should listen to the economic experts and dramatically increase immigration levels (at least 500,000 per year). Stop listening to the loser racist trash .
AK Jen - Russia is no longer a communist nation. Stop spouting that - it makes you look stupid.	AK Jen - Russia is no longer a communist nation. Stop spouting that - it makes you look stupid .
The "human rights" squad need a swift kick in the derriere.	The "human rights" squad need a swift kick in the derriere .
Rabidly anti-Canadian troll .	Rabidly anti-Canadian troll .
Lord have Mercy on us, Trump is running amok	Lord have Mercy on us, Trump is running amok.
You are a nut ball	You are a nut ball
Lives in Thailand?	Lives in Thailand?
So like every other racist, he's a hypocrite	So like every other racist, he's a hypocrite
Terry Stahlman is a worthless piece of human excrement!	Terry Stahlman is a worthless piece of human excrement!
People who are anti-immigration are weak, lying, racist whiners .	People who are anti-immigration are weak, lying, racist whiners.
Some people don't need dope to be dopey.	Some people don't need dope to be dopey .
But that's what you get now for minimum wage. Increasing that to a 'livable wage' for the exact same people is stupid!	But that's what you get now for minimum wage. Increasing that to a 'livable wage' for the exact same people is stupid !
They can't, it's jammed with an overload of their hero's excrement .	They can't, it's jammed with an overload of their hero's excrement.
holy fuck you troglodytes cant even handle the simple act of meming	holy fuck you troglodytes cant even handle the simple act of meming
Meanwhile Taxed is now complaining his Hyundai Santa Fe is a piece of crap	Meanwhile Taxed is now complaining his Hyundai Santa Fe is a piece of crap
Only in that sick and twisted brain stem of yours.	Only in that sick and twisted brain stem of yours.

Table 3: Examples from the competition test dataset of differences between the annotations and the predictions from BERT-toxic-CRF-VAT model. The toxic spans are highlighted.

part of the sequence of words. Samples like "You are a nut ball" detected only as "You are a nut ball" or "So like every other racist, he's a hypocrite" marked by the model as "So like every other racist, he's a hypocrite" can be perceived as likely errors even for human annotators.

## 6 Conclusions and Future Work

In this paper, several Transformer-based models (i.e., BERT and RoBERTa) were tested together with Virtual Adversarial Training to increase their robustness for identifying toxic spans from textual information. Our experiments argue that applying VAT increases performance and that domain-specific models have higher performance when compared to larger general models.

In terms of future work, we plan to experiment with self-supervised adversarial training (Chen et al., 2020) to improve the robustness of our models. As we noticed in this dataset too, online users find clever ways to hide offensive and toxic expressions. Adversarial training can be effectively employed to detect these attempts and a study of its impact on offensive and hate speech classifiers is worth pursuing as follow-up leads.

## References

- Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42.
- ME Aragón, H Jarquín, M Montes-y Gómez, HJ Escalante, L Villaseñor-Pineda, H Gómez-Adorno, G Bel-Enguix, and JP Posadas-Durán. 2020. Overview of mex-a3t at iberlef 2020: Fake news and aggressiveness analysis in mexican spanish. In *Notebook Papers of 2nd SEPLN Workshop on Iberian Languages Evaluation Forum (IberLEF)*, Malaga, Spain.
- Andrei-Marius Avram, Dumitru-Clementin Cercel, and Costin Chiru. 2020. Upb at semeval-2020 task 6: Pretrained language models for definition extraction. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 737–745.
- Kejiang Chen, Yuefeng Chen, Hang Zhou, Xiaofeng Mao, Yuhong Li, Yuan He, Hui Xue, Weiming Zhang, and Nenghai Yu. 2020. Self-supervised adversarial training. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2218–2222. IEEE.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Oguzhan Gencoglu. 2020. Cyberbullying detection with fairness constraints. *IEEE Internet Computing*.
- Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.
- Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1923–1933.
- Marius Ionescu, Andrei-Marius Avram, George-Andrei Dima, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. Upb at fincausal-2020, tasks 1 & 2: Causality analysis in financial documents using pre-trained language models. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 55–59.
- Mika Juuti, Tommi Gröndahl, Adrian Flanagan, and N Asokan. 2020. A little goes a long way: Improving toxic language classification despite data scarcity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 2991–3009.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.
- Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2741–2749.

- Priyanshu Kumar and Aadarsh Singh. 2020. Nutcracker at wnut-2020 task 2: Robustly identifying informative covid-19 tweets using ensembling and adversarial training. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 404–408.
- Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.
- Kyounghee Kwon and Anatoliy Gruzd. 2017. Is offensive commenting contagious online? examining public vs interpersonal swearing in response to donald trump’s youtube campaign videos. *Internet Research*, 27(4):991–1010.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.
- Tagrid Lemenager, Miriam Neissner, Anne Koopmann, Iris Reinhard, Ekaterini Georgiadou, Astrid Müller, Falk Kiefer, and Thomas Hillemacher. 2021. Covid-19 lockdown restrictions and online media consumption in germany. *International journal of environmental research and public health*, 18(1):14.
- Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fernández, Silvio Amir, Luis Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.
- Jiaxiang Liu, Xuyi Chen, Shikun Feng, Shuohuan Wang, Xuan Ouyang, Yu Sun, Zhengjie Huang, and Weiyue Su. 2020. Kk2018 at semeval-2020 task 9: Adversarial training for code-mixing sentiment classification. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 817–823.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.
- Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. 2020. Overview of the hasoc track at fire 2020: Hate speech and offensive language identification in tamil, malayalam, hindi, english and german. In *Forum for Information Retrieval Evaluation*, pages 29–32.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2015. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*.
- Sergey Morzhov. 2020. Avoiding unintended bias in toxicity classification with neural networks. In *2020 26th Conference of Open Innovations Association (FRUCT)*, pages 314–320. IEEE.
- Andrei Paraschiv and Dumitru-Clementin Cercel. 2019. Upb at germeval-2019 task 2: Bert-based offensive language classification of german tweets. In *KONVENS*.
- Andrei Paraschiv, Dumitru-Clementin Cercel, and Mihai Dascau. 2020. Upb at semeval-2020 task 11: Propaganda detection with domain-specific trained bert. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1853–1857.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.
- Khoury Richard and Larochelle Marc-André. 2020. Generalisation of cyberbullying detection. *arXiv preprint arXiv:2009.01046*.
- Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2020. Better robustness by more coverage: Adversarial training with mixup augmentation for robust fine-tuning. *arXiv preprint arXiv:2012.15699*.
- Julia Maria Struß, Melanie Siegel, Josef Ruppenhofer, Michael Wiegand, Manfred Klenner, et al. 2019. Overview of germeval task 2, 2019 shared task on the identification of offensive language. In *KONVENS*.

- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- Mircea-Adrian Tanase, Dumitru-Clementin Cercel, and Costin Chiru. 2020a. Upb at semeval-2020 task 12: Multilingual offensive language detection on social media by fine-tuning a variety of bert-based models. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2222–2231.
- Mircea-Adrian Tanase, George-Eduard Zaharia, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020b. Detecting aggressiveness in mexican spanish social media content by fine-tuning transformer-based models. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020) co-located with 36th Conference of the Spanish Society for Natural Language Processing (SEPLN) 2020*, pages 236–245.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447.

# NLRG at SemEval-2021 Task 5: Toxic Spans Detection Leveraging BERT-based Token Classification and Span Prediction Techniques

**Gunjan Chhablani\***

Dept. of CS&IS  
BITS Pilani, Goa, India

chhablani.gunjan@gmail.com

**Abheesht Sharma\***

Dept. of CS&IS  
BITS Pilani, Goa, India

f20171014@goa.bits-pilani.ac.in

**Harshit Pandey\***

Dept. of CS  
Pune University, India

hp2pandey1@gmail.com

**Yash Bhartia**

Dept. of CS&IS  
BITS Pilani, Goa, India

f20190151@goa.bits-pilani.ac.in

**Shan Suthaharan**

Dept. of CS  
UNC-Greensboro, NC, USA

s\_suthah@uncg.edu

## Abstract

Toxicity detection of text has been a popular NLP task in the recent years. In SemEval-2021 Task-5 Toxic Spans Detection, the focus is on detecting toxic spans within English passages. Most state-of-the-art span detection approaches employ various techniques, each of which can be broadly classified into Token Classification or Span Prediction approaches. In our paper, we explore simple versions of both of these approaches and their performance on the task. Specifically, we use BERT-based models - BERT, RoBERTa, and SpanBERT for both approaches. We also combine these approaches and modify them to bring improvements for Toxic Spans prediction. To this end, we investigate results on four hybrid approaches - Multi-Span, Span+Token, LSTM-CRF, and a combination of predicted offsets using union/intersection. Additionally, we perform a thorough ablative analysis and analyze our observed results. Our best submission - a combination of SpanBERT Span Predictor and RoBERTa Token Classifier predictions - achieves an  $F_1$  score of 0.6753 on the test set. Our best post-eval  $F_1$  score is 0.6895 on intersection of predicted offsets from top-3 RoBERTa Token Classification checkpoints. These approaches improve the performance by 3% on average than those of the shared baseline models - RNNLS and SpaCy NER.

## 1 Introduction

Offensive language can include various categories such as threats, vilification, insults, calumny, discrimination and swearing (Pavlopoulos et al., 2019). Detection of such language is necessary for ease of moderation of content on social media. Despite their popularity, toxicity detection tasks have focused majorly on sequence classification, rather

\* Equal contribution. Author ordering determined by coin flip.

than sequence tagging. Finding which spans make a comment or document toxic in nature is crucial in explaining the reasons behind their toxicity. Additionally, such attributions would allow for more efficient semi-automated quality-based moderation of content, especially for verbose documents, in comparison to quantitative toxicity scores.

In SemEval-2021 Task-5, Pavlopoulos et al. (2021) provide a dataset of 10k English texts filtered from Civil Comments (Borkan et al., 2019) dataset. Each text is crowd-annotated with character offsets that make the text toxic. The task is to predict these character offsets given the text. The work presented in this paper aims to provide a comprehensive analysis of simple Token Classification (TC) and Span Prediction (SP) methods across multiple BERT-based models - BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and SpanBERT (Joshi et al., 2020). Additionally, we experiment with a few hybrid approaches - Multi-Span (MSP), where the model is trained on multiple spans simultaneously; Span+Token (SP-TC), where the model is trained on both kinds of tasks simultaneously; LSTM-CRF (LC), which uses a LSTM and CRF layer on top of BERT-based models; and a combination of predicted offsets for above techniques using union/intersection. In Section 2, we perform a compendious literature survey. Section 3 elucidates our approach, including the modelling aspect, the various variants of the base model, and the different Hybrid Systems. In Section 4, we describe our experimental setup and hyperparameters used for our methods. Lastly, in Section 5 we analyze our results and perform ablative analysis on our systems.

## 2 Background

Before the advent in research pertaining to toxic texts, Warner and Hirschberg (2012) modeled hate

speech as a word sense disambiguation problem where SVM was used for classification of data. Mehdad and Tetreault (2016) used RNN Language Model with character and token based methods to classify the text. Recently, however, toxic text detection has garnered a lot of attention (Nobata et al., 2016; Park and Fung, 2017; Pavlopoulos et al., 2017; Wulczyn et al., 2017). The increase in offensive language research can partly be credited to various workshops such as Abusive Language Online<sup>1</sup> (Waseem et al., 2017), as well as other fora, such as GermEval for German texts,<sup>2</sup> or TRAC (Kumar et al., 2018) and Kaggle challenges<sup>3</sup>.

Hanu and Unitary team (2020) introduced Detoxify, a comment detection library modeled using HuggingFace’s transformers (Wolf et al., 2020) to identify inappropriate or harmful text online as a result of participation in three such challenges. In a contemporary work, Pavlopoulos et al. (2020) discuss context requirement for toxicity detection.

In SemEval 2020-Task 11 (Da San Martino et al., 2020), the first sub-task - Span Identification - aims at detecting the beginning and the end offset for the propaganda spans in news articles. This sub-task is similar to SemEval 2021-Task 5. The proposed approaches for the sub-task can be broadly classified into Span Prediction or Token Classification. Most teams use multi-granular transformer-based systems for token classification/sequence tagging (Khosla et al., 2020; Morio et al., 2020; Patil et al., 2020). Inspired by Souza et al. (2019), Jurkiewicz et al. (2020) use RoBERTa-CRF based systems. Li and Xiao (2020) use a variant of SpanBERT span prediction system.

### 3 Models

#### 3.1 Token Classification Models

##### 3.1.1 Baseline Models

From the models already provided with the dataset, we use RNNSL and SpaCy NER Tagging baselines for token-wise classification.

RNNSL model is a combination of a single Bi-LSTM layer with a randomly initialized embedding layer. It uses a three-label classification task for each word in the sentence. The labels used are: *special token*, *non-toxic word*, and *toxic word*. For

<sup>1</sup><https://sites.google.com/site/abusivelanguageworkshop2017/>

<sup>2</sup><https://projects.fzai.h-da.de/iggsa/>

<sup>3</sup>Jigsaw Toxic Comment Classification Challenge

each word, the corresponding offsets are added to the predicted spans. A word with containing any toxic offset is marked as toxic during training.

SpaCy NER Tagging model is an NER classifier built on SpaCy Language Models. It is used to predict the entities which are labelled as *TOXIC* in the text using the spans provided.

#### 3.1.2 BERT-based Token Classification Models

These models comprise a BERT-based model and a classification layer over each final token embedding which predicts whether a token is toxic or not. Based on these classifications, we add the offsets for those tokens (not words) which are marked as toxic by the model. Figure 1a represents a Token Classification Model.

### 3.2 Span Prediction Models

#### 3.2.1 BERT-based Span Prediction Models

We use the BERT-based Span Prediction (Figure 1c) models based on Extractive Question Answering systems similar to work on SQuAD (Rajpurkar et al., 2016) and MRQA (Fisch et al., 2019). In these systems, the output at each token is a start logit and an end logit denoting whether that token is a start token or an end token of the span, depending on the softmax value. Since the Toxic Spans text can have multiple toxic spans, we take different contiguous spans from the given offsets, and make several ‘samples’ out of the example. Each span becomes an ‘answer’ for the particular text sample. We use the word ‘*offense*’ as a dummy question. Thus, each contiguous span leads to one ‘sample’ for every example (Table 1).

Text		Spans
...an idiot - just an embarrassingly uninformed, ignorant,...		idiot, ignorant
Question	Context	Answer
offense	...an idiot - just an embarrassingly uninformed, ignorant,...	idiot
offense	...an idiot - just an embarrassingly uninformed, ignorant,...	ignorant

Table 1: Conversion of Toxic Spans example to samples for single-span Span Prediction.

We store the start index of the text, similar to the SQuAD (Rajpurkar et al., 2016) dataset, and process the data to provide start and end token positions during training. The classifier layer on top of the encoder embeddings performs a binary classification task for start and end positions. A

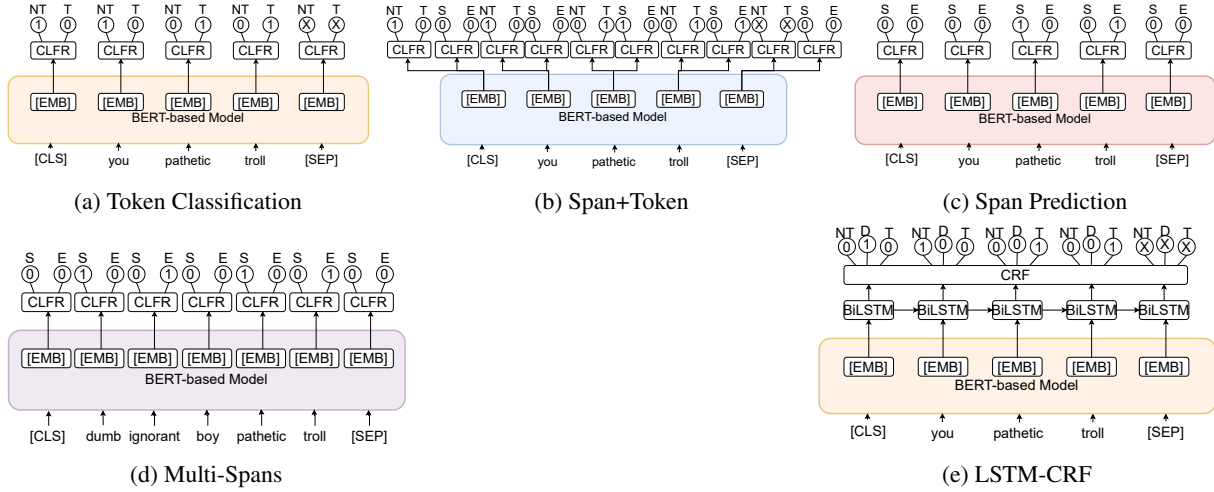


Figure 1: BERT-based Approaches\*

\* CLFR = Classifier, [EMB] = Token Embedding, NT = Non-Toxic, T = Toxic, D = Dummy, X = Don't Care, S = Start, E = End.

span is scored using the sum of predicted start and end logits. From top-K start and end logits, valid predicted answer spans<sup>4</sup> are chosen during post-processing. A union of all the corresponding offsets is taken to give the final prediction for the example. A threshold is learned on the span scores using the resulting dev set  $F_1$  score on offsets, which is then used for test set prediction. All spans with score above threshold are considered to be toxic spans.

### 3.3 Hybrid Systems

#### 3.3.1 Multi-Spans

In Section 3.2, we allow each context to have multiple single-span answers during training. This is counter-intuitive, as the model is only trained to handle a single-span at a time, and expected to predict multiple single-spans during prediction. Two toxic spans in text are equally important to predict, and thus, should not be shown at different times during training. To mitigate this issue, we try an approach which we refer to as the ‘Multi-Spans’ (MSP) approach. Here, we take all the ground start and end token positions during training, and use Binary Cross Entropy on each of the start/end logits. This essentially treats the task as a multi-label classification problem. Hence, during training, all the ground spans are used in the same iteration with the example, and only one ‘sample’ per example is generated. Figure 1d depicts a representation of the system. Note that two tokens - *dumb* and *pathetic* are marked as the start token. Similarly, both *ignorant* and *troll* are marked as the end token.

<sup>4</sup>Valid spans are those which have end index greater than start index, and length less than a maximum span length.

#### 3.3.2 LSTM-CRF

A recently popular approach in Named-Entity Recognition tasks has been to use Conditional Random Fields (CRF) with BERT-based models. Inspired by the CRF-based approaches (Souza et al., 2019; Jurkiewicz et al., 2020), we use BERT-based models with a single BiLSTM layer and a CRF layer. During training, the CRF loss is used and during prediction, Viterbi Decoding is performed. Though CRF is generally used for word-level classification, we do not mask inner and end tokens for a word as it degrades dev set performance for our systems. Hence, all the tokens of a word are considered for classification.

#### 3.3.3 Spans+Token

For this system, we use a combination of the two tasks - Token Classification and single-span Span Prediction. We use two classification layers on the token-wise embeddings - one for start and end prediction, and the other for token classification. Training is done simultaneously on both tasks, and the cross-entropy loss for each classifier is weighted. The overall loss is given as:

$$L(\hat{s}, \hat{e}, \hat{p}, s, e, p) = - \sum_t \hat{p}_t \log p_t - \frac{(\sum_t \hat{s}_t \log s_t + \sum_t \hat{e}_t \log e_t)}{2}$$

where  $s_t, e_t$ , and  $p_t$  are labels for start, end and token classifiers for token  $t$ , while  $\hat{s}_t, \hat{e}_t$  and  $\hat{p}_t$  are predictions. This is done to equally scale both SP and TC task losses. During prediction, we consider top-K start and end scores. From the valid spans,

the score is calculated as the average of start and end logit scores, as well as the mean of toxicity logits over the span under consideration. The score is given as:

$$S(i_s, i_e) = \frac{\hat{s}_{i_s} + \hat{e}_{i_e}}{2} + \frac{\sum_{k=i_s}^{i_e} \hat{t}_k}{e - s + 1}$$

where  $i_s$  and  $i_e$  are start and end indices,  $\hat{s}_{i_s}$  and  $\hat{e}_{i_e}$  are start and end logits at those indices, and  $\hat{t}_k$  is toxicity logit at index  $k$ . A threshold, similar to Section 3.2 is tuned on the dev set. The predicted offsets taken from the predicted spans are considered to be toxic.

### 3.3.4 Combination of Offset Predictions

Chen et al. (2017) proposed using the predictions from top few checkpoints and averaging the results to achieve better classification scores. Based on a similar line of thought, we also combine the predicted spans for various checkpoints of a model, as well as across different models using union or intersection.

## 4 Experimental Setup<sup>5,6</sup>

### 4.1 Hardware Requirements

The training and the evaluation of systems was performed on Google Colab’s free GPU (NVIDIA K80/P100). The training time varies with the models. For each model, it is around 4-6 hours, which is well-within the 12 hour limit of Colab.

### 4.2 Models & Hyperparameters

For RNNSL, a Keras-based BiLSTM model is provided. We use a max length of 192, batch size of 32 and a dropout of 0.1. The training is done using Adam Optimizer with early stopping (*patience\_period* = 3), which in our case halts at 5 epochs. The embedding/hidden\_state size used is 200. A threshold is used to classify a word as toxic on the predicted toxic word probability. This threshold is tuned on the trial dataset. For SpaCy, the *en\_core\_web\_sm* model is used with 30 iterations.

For all BERT-based models, we use Hugging-Face’s transformers (Wolf et al., 2020) in PyTorch. For CRF, we use the pytorch-crf (Kurniawan, 2018) library. We use a batch size of 4, train for 3 epochs,

<sup>5</sup>Our code can be found at: <https://github.com/gchhablani/toxic-spans-detection>.

<sup>6</sup>We also use Integrated Gradients to understand what the models focus on. For discussion, see Appendix B.

use a linear learning rate decay, and an AdamW optimizer with a weight decay of 0.01. The initial learning rate is  $2e-5$ . During tokenization, the maximum length allowed is 384, with the exception of RoBERTa Span+Token where it is 512. We use *LARGE* models for all - BERT, RoBERTa and SpanBERT, unless otherwise specified.

For Token Classification, we add a label for the *[CLS]* token if the percentage of toxic offsets in text is greater than 30% in order to provide a proxy text classification objective for the system. For span-based models, the K used for top-K start and top-K end logit selection is 20, and the maximum allowed answer length is 30 tokens. For LSTM-CRF systems, a dummy label is used for the *[CLS]* token, while the prediction mask for other special tokens is set to 0. A dropout of 0.2 is used. For Span Prediction systems, the overlapping stride is set to 128.

The training dataset used is *tsd\_train.csv* and the dev set used is *tsd\_trial.csv* file, unless otherwise specified. For all systems, we evaluate the  $F_1$  scores using the provided script on the checkpoints which give the lowest dev set loss.

## 5 Results and Analysis

In favor of brevity, for this section, we use the following abbreviations: BT=BERT, RBTa=RoBERTa, SBT=SpanBERT, SP=Span Prediction, TC=Token Classification, MSP=Multi-Span, LC=LSTM-CRF, B=Base, TBT=ToxicBERT, TRBTa=ToxicRoBERTa, TT=Trained on Train+Trial, (x,∩)=Intersection of offsets from x-best checkpoints, (x,∪)=Union of offsets from x-best checkpoints.

In Table 2, we mention scores for our approaches. The scores are evaluated are performed after the evaluation phase, using the hyperparameters mentioned in Section 4.2. We observe that the highest score is obtained by SBT-TC (0.6856). The baseline scores (RNNSL/SpaCy) are good ( $\approx 0.65$ ) considering that these models are not pre-trained. Notably, SP systems perform worse than their TC counterparts. A good reason could be the self-attention used in BERT-based models. Since the interaction is between tokens, and not spans, it is expected that each token is well represented and less consideration will be given to the span representation around a single token. The reason why SBT-TC performs best out of all the *LARGE* models could be the random-spans Masked Language



Model	Train $F_1$	Trial $F_1$	Test $F_1$
RNNSL	0.5904	0.5904	0.6514
SpaCy	0.6282	0.5729	0.6573
BT-TC	0.6944	0.6942	0.6781
RB-Ta-TC	0.6791	0.6769	0.6834
SBT-TC	0.6873	0.6789	<b>0.6856</b>
BT-SP	0.6639	0.6465	0.6663
RB-Ta-SP	0.6401	0.6386	0.6665
SBT-SP	0.6432	0.6212	0.6561
BT-MSP	0.5218	0.4941	0.5406
RB-Ta-MSP	0.5056	0.4886	0.5244
SBT-MSP	0.5190	0.5004	0.5084
BT-SP-TC	0.6676	0.6214	0.6186
RB-Ta-SP-TC	0.6395	0.6101	0.5901
SBT-SP-TC	0.6608	0.6491	0.5959
BT-LC	0.6887	0.6843	0.6835
RB-Ta-LC	0.7236	0.6861	0.6787
SBT-LC	0.7200	0.6982	0.6801

Table 2:  $F_1$  scores for our approaches (Post-Eval).

Modeling used in its pre-training. However, BERT and RoBERTa take over for other approaches.

LSTM-CRF approaches perform as good as Token Classification approaches, and BT-LC achieves the second highest score (0.6835). MSP performs poorly, in contrast to what is expected. Multi-Span Extraction is still an active problem in Deep NLP with only a few recent works (Segal et al., 2020; Yang et al., 2020) on it, which still incorporate sequence tagging approaches. Spans+Token approaches perform better than Multi-Span, but are worse than both TC and SP approaches across all BERT-based models.

Lastly, from combined checkpoint predictions

Combination	Test $F_1$
RB-Ta-TC(3, $\cup$ )	0.6765
RB-Ta-TC(3, $\cap$ )	<b>0.6895</b>
SBT-SP(3, $\cup$ )	0.5879
SBT-SP(3, $\cap$ )	0.6585
RB-Ta-TC(3, $\cup$ ) $\cup$ SBT-SP	0.6573
RB-Ta-TC(3, $\cup$ ) $\cap$ SBT-SP	0.6765
RB-Ta-TC $\cup$ SBT-SP(3, $\cup$ )	0.5840
RB-Ta-TC $\cap$ SBT-SP(3, $\cup$ )	0.6883

Table 3:  $F_1$  scores for combined predictions.

(Table 3), we get out best scoring system - RB-Ta-TC(3, $\cap$ ) - which achieves a score of 0.6895. However, our best official submission<sup>7</sup> was a variant of the third best combination - RB-Ta-TC(3, $\cup$ ) $\cap$ SBT-SP (0.6765). It is also observed that intersection ap-

<sup>7</sup>The most significant of our official submission scores are present in Appendix A.

proaches perform better than corresponding union and single checkpoints approaches, while union approaches perform worse than single checkpoints. This means that the individual checkpoints are predicting some extra offsets to be toxic.

## 5.1 Ablative Analysis

Model	Train $F_1$	Trial $F_1$	Test $F_1$
TBT-TC	0.6753	0.6628	0.6792
TRB-Ta-TC	0.7244	0.6954	0.6773
TBT-SP	0.6638	0.6560	0.6584
TRB-Ta-SP	0.6475	0.6358	0.6746
BT-B-TC	0.6966	0.6746	<b>0.6881</b>
RB-Ta-B-TC	0.6641	0.6482	0.6834
BT-B-SP	0.6605	0.6434	0.6611
RB-Ta-B-SP	0.6481	0.6464	0.6661
RNNSL-TT	0.6844	0.6882	0.6259
RB-Ta-TC-TT	0.7707	0.7788	0.6823
SBT-SP-TT	0.7116	0.7092	0.6669

Table 4:  $F_1$  scores for ablative approaches.<sup>8</sup>

In Table 4, we present results on TBT<sup>8</sup> and TRB-Ta<sup>9</sup> for TC and SP approaches. These are *BASE* models fine-tuned on the Civil Comments Dataset. Since the Toxic Spans dataset has similar text data, we expect these models to perform better than *BASE* models. We observe that TBT-TC and TRB-Ta-SP perform slightly better than BT-TC and RB-Ta-SP, despite being *BASE* models. Also, BT-SP and RB-Ta-TC are only slightly better than their ‘Toxic counterparts.

Yet, in comparison, *BASE* models - BT-B and RB-Ta-B, without any multi-stage pre-training perform better than their ‘Toxic’ counterparts, and are comparable, if not better than their *LARGE* counterparts. This means that there not enough data for *LARGE* models, and hence, they tend to overfit. However, the reasons behind worse performance of ‘Toxic’ systems is unclear.

We also evaluate scores for a few systems on the test set after 3 epochs of training on both train and trial data (-TT). We observe that the performance on both train and trial datasets increases significantly ( $\approx 7$ -10%), showing that these datasets have similar distribution. However, the performance on test decreases for RB-Ta-TC-TT and RNNSL-TT in comparison to the Table 2, which shows that test set distribution might be slightly different for TC task. For SBT-SP-TT, we see a slight increase, showing

<sup>8</sup><https://huggingface.co/unitary/toxic-bert>

<sup>9</sup><https://huggingface.co/unitary/unbiased-toxic-roberta>

scope of improvement for SP systems with more data.

Lastly, we evaluate the token-based predictions and span-based predictions for SBT SP-TC separately. Surprisingly, token predictions achieve a  $F_1$  score of 0.6522 on the test set, which is much better than using both token and spans (0.5959). However, for span-based predictions, we only achieve an  $F_1$  score of 0.1510. This means that the system is focusing heavily on token-based-predictions. Hence, we need to re-evaluate our architectural decisions in order to successfully incorporate both token and spans together.

## 6 Conclusion

Based on our results and analysis, we conclude that Token Classification systems have an edge over Span Prediction methods on this task. *BASE* models perform better than *LARGE* models in either of the approaches, which could imply need for more data to train *LARGE* models. Our Multi-Span approach performs poorly, but Span+Token approach shows some promise and we need to re-evaluate our architectural choices. The reason why ToxicBERT/ToxicRoBERTa perform worse than *BASE* models is also an avenue for further analysis. Finally, our individual BERT-based models tend to predict extra offsets for the task. While checkpoint ensembling using intersection is a good way to address this issue, we will explore other remedies in a future work.

## Acknowledgments

We would like to acknowledge the help and moral support provided to us by Rajaswa Patil<sup>10</sup> and Somesh Singh<sup>11</sup>. We would also like to express our gratitude to our colleagues at the Language Research Group (LRG)<sup>12</sup>, who have been with us at every stepping stone.

## References

- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Hugh Chen, Scott Lundberg, and Su-In Lee. 2017. [Checkpoint ensembles: Ensemble methods from a single training process](#). *CoRR*, abs/1710.03282.

<sup>10</sup><https://rajaswa.github.io/>

<sup>11</sup><https://someshsingh22.github.io/>

<sup>12</sup><https://lrg.saidl.in/>

- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. [SemEval-2020 task 11: Detection of propaganda techniques in news articles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, Barcelona (online). International Committee for Computational Linguistics.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

- Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. [MRQA 2019 shared task: Evaluating generalization in reading comprehension](#). In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

- Laura Hanu and Unitary team. 2020. [Detoxify](#). Github. <https://github.com/unitaryai/detoxify>.

- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. [Spanbert: Improving pre-training by representing and predicting spans](#).

- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. [ApplicaAI at SemEval-2020 task 11: On RoBERTa-CRF, span CLS and whether self-training helps them](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1415–1424, Barcelona (online). International Committee for Computational Linguistics.

- Sopan Khosla, Rishabh Joshi, Ritam Dutt, Alan W Black, and Yulia Tsvetkov. 2020. [LTIatCMU at SemEval-2020 task 11: Incorporating multi-level features for multi-granular propaganda span identification](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1756–1763, Barcelona (online). International Committee for Computational Linguistics.

- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. [Captum: A unified and generic model interpretability library for pytorch](#).

- Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. [Benchmarking aggression identification in social media](#). In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

- Kemal Kurniawan. 2018. Pytorch-crf. <https://github.com/kmkurn/pytorch-crf>.
- Jinfen Li and Lu Xiao. 2020. [syraproa at SemEval-2020 task 11: BERT-based models design for propagandistic technique and span detection](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1808–1816, Barcelona (online). International Committee for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Yashar Mehdad and Joel Tetreault. 2016. [Do characters abuse more than words?](#) In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, Los Angeles. Association for Computational Linguistics.
- Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. [Hitachi at SemEval-2020 task 11: An empirical study of pre-trained transformer family for propaganda detection](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1739–1748, Barcelona (online). International Committee for Computational Linguistics.
- Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. [Abusive language detection in online user content](#). In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 145–153, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Ji Ho Park and Pascale Fung. 2017. [One-step and two-step classification for abusive language detection on Twitter](#). In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45, Vancouver, BC, Canada. Association for Computational Linguistics.
- Rajaswa Patil, Somesh Singh, and Swati Agarwal. 2020. [BPGC at SemEval-2020 task 11: Propaganda detection in news articles with multi-granularity knowledge sharing and linguistic features based ensemble learning](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1722–1731, Barcelona (online). International Committee for Computational Linguistics.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, Juli Bakagianni, and Ion Androutsopoulos. 2017. [Improved abusive comment moderation with user embeddings](#). In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*, pages 51–55, Copenhagen, Denmark. Association for Computational Linguistics.
- John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. [Toxicity detection: Does context really matter?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.
- John Pavlopoulos, Nithum Thain, Lucas Dixon, and Ion Androutsopoulos. 2019. [ConvAI at SemEval-2019 task 6: Offensive language identification and categorization with perspective and BERT](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 571–576, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Sahana Ramnath, Preksha Nema, Deep Sahni, and Mitesh M. Khapra. 2020. [Towards interpreting BERT for reading comprehension based QA](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3236–3242, Online. Association for Computational Linguistics.
- Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. [A simple and effective model for answering multi-span questions](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080, Online. Association for Computational Linguistics.
- Fábio Souza, Rodrigo Frassetto Nogueira, and Roberto de Alencar Lotufo. 2019. [Portuguese named entity recognition using BERT-CRF](#). *CoRR*, abs/1909.10649.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. [Axiomatic attribution for deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17*, pages 3319–3328. JMLR.org.
- William Warner and Julia Hirschberg. 2012. [Detecting hate speech on the world wide web](#). In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. Association for Computational Linguistics.
- Zeerak Waseem, Wendy Hui Kyong Chung, Dirk Hovy, and Joel Tetreault, editors. 2017. *Proceedings of the First Workshop on Abusive Language Online*. Association for Computational Linguistics, Vancouver, BC, Canada.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen,

Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#).

Junjie Yang, Zhuosheng Zhang, and Hai Zhao. 2020. [Multi-span style extraction for generative reading comprehension](#).

## A Official Submissions

During the evaluation period, we performed a ‘cleaning’ of the data by removing starting/trailing whitespace and punctuation characters in spans. Additionally, we include those partial words in spans which had more than half the number of characters in the span, and discard remaining partial words from spans. We considered this version of the *tsd\_train.csv* and *tsd\_trial.csv* to be ‘clean train’ and ‘clean trial’, respectively. During the post-eval period, we found out potential issues with the cleaning, and thus, we use original files. Additionally, since the distribution of *tsd\_test.csv* is expected to be similar to *tsd\_train.csv* and *tsd\_trial.csv*, the scores are much better for models trained on *tsd\_train.csv* file instead of *clean\_train.csv*. However, some of our official submissions were from systems trained on the ‘clean train’ data. Keeping that in mind, we report our official scores for our top-few approaches in Table 5.

Model	Trained On	Test $F_1$
RNNSL	Train+Trial	0.6446
SpaCy	Train+Trial	0.6470
RNNSL $\cup$ SpaCy	Train+Trail	0.6510
RBTa-TC	Clean Train	0.6270
RBTa-TC(3, $\cup$ )	Clean Train	0.6469
SBT-SP	Train	0.6631
RBTa-TC(3, $\cup$ ) $\cap$ SBT-SP	Clean Train, Train	<b>0.6753</b>

Table 5: Official Submission Scores

## B Integrated Gradients

We use Integrated Gradients(Sundararajan et al., 2017) from the Captum(Kokhlikyan et al., 2020) library for qualitative analysis of predictions for the SpanBERT-SP, and the RoBERTa-TC models. We calculate Integrated Gradients of the targets with respect to the embedding layer outputs. The

Riemann Right numerical approximation method is used, with  $n\_steps=50$ . Following Ramnath et al. (2020), we calculate token-wise importance distributions and word-wise distributions for a few examples. We refer the paper to the reader for more details.

For the Token Classification model, the targets are softmax outputs of toxicity logits of those tokens which the model predicts to be toxic, with a score greater than 0.5. For all such toxicity logits as targets, we calculate attributions with respect to the embedding layer outputs for all the tokens, and average them to get token-wise importance scores. For the Span Prediction model, we find start and end indices for all the predicted spans, and calculate respective attributions, add them, and then average them to get token-wise importance scores.

**Text:** offense See a shrink you pathetic troll .

**Ground Spans:** [ 'pathetic troll' ]  
**Predicted Spans:** [ 'pathetic troll' ]

(a) SpanBERT Span Prediction

**Text:** See a shrink you pathetic troll.

**Ground Spans:** [ 'pathetic troll' ]  
**Predicted Spans:** [ 'pathetic', 'troll' ]

(b) RoBERTa Token Classification

Figure 2: Qualitative Example of Attributions - Example 1

**Text:** offense Stupid is as stupid does Gump was right

**Ground Spans:** [ 'Stupid', 'stupid' ]  
**Predicted Spans:** [ 'Stupid is as stupid' ]

(a) SpanBERT Span Prediction

**Text:** Stupid is as stupid does Gump was right

**Ground Spans:** [ 'Stupid', 'stupid' ]  
**Predicted Spans:** [ 'Stupid', 'stupid' ]

(b) RoBERTa Token Classification

Figure 3: Qualitative Example of Attributions - Example 2

We observe in Figure 2a that the Span Prediction model performs correct prediction. However, on average, the word ‘shrink’ gets higher importance

**Text** : Why does this author think she can demand, or is owed anything from either of these two people? One guy is a goon, the other is illiterate. They aren't law makers, teachers, or in any kind moral authority position. They are entertainers who get punched for her pleasure, and will likely live out their days mentally debilitated from the repeated blows to the head.

Do we get to comb deeply through this authors personal history and determine all the groups she owes apologies or explanations to? Why not? As an opinion maker in a national news paper and instructor of young people, she has far, far more influence on Canadians than two ignorant punchies. The arrogance of these pseudo-intellectual academics is astounding. Since they are so enlightened and pure, YOU owe THEM an explanation and an apology as to why you're so dumb and ignorant.

**Ground Spans:** [dumb]

BT-B-SP	[]
BT-B-TC	[dumb, ignorant]
BT-LC	[dumb, ignorant]
BT-MSP	[dumb]
BT-SP	[]
BT-TC	[dumb, ignorant]
BT-SP-TC	[dumb and ignorant]
RBTa-TC(3, $\cap$ )	[dumb, ignorant]
RBTa-TC $\cap$ SBT-SP(3, $\cup$ )	[dumb, ignorant]
SBT-SP(3, $\cap$ )	[]
RBTa-TC(3, $\cup$ ) $\cap$ SBT-SP	[]
RBTa-TC(3, $\cup$ )	[go, dumb, ignorant]
RBTa-TC $\cup$ SBT-SP(3, $\cup$ )	[dumb and ignorant]
SBT-SP(3, $\cup$ )	[dumb and ignorant]
RBTa-TC(3, $\cup$ ) $\cup$ SBT-SP	[go, dumb, ignorant]
RNNSL	[ignorant, dumb, ignorant]
RNNSL-TT	[goon, ignorant, dumb, ignorant]
RBTa-B-SP	[]
RBTa-B-TC	[dumb]
RBTa-LC	[on, ignorant, dumb, ignorant]
RBTa-MSP	[]
RBTa-SP	[]
RBTa-TC	[dumb, ignorant]
RBTa-SP-TC	[ignorant, dumb and ignorant]
RBTa-TC-TT	[dumb, ignorant]
SpaCy	[ignorant]
SBT-LC	[ignorant, dumb, ignorant]
SBT-MSP	[dumb and ignorant]
SBT-SP	[]
SBT-SP-TT	[dumb and ignorant]
SBT-TC	[ignorant, dumb, ignorant]
SBT-SP-TC	[ignorant, dumb and ignorant]
TBT-SP	[]
TBT-TC	[ignorant]
TRBTa-SP	[]
TRBTa-TC	[dumb, ignorant]

Table 6: The prediction output of the models for an example in the test set.

than *'pathetic troll'*. This is in contrast with Figure 2b where the Token Detection model misses out on space (because it only considers tokens) and focuses more on the words *'pathetic'*, *'troll'*. However, the word *'shrink'* seems to be important in both cases. This means that while Token Classification models perform better, there are cases which are missed by these approaches. Additionally, some words outside of the span may contribute to toxicity of a particular span. We will be analyzing such words in a future work.

## C Model Predictions

The predictions of the various systems for one example that is present in the test set, are listed in Table 6. The examples provide the following intuition about the data and the systems:

- The spaces in between the words are, predictably, ignored by the token based models. Moreover, the conjunctives like 'and' are ignored as well. This means that additional post-processing of the data will lead to improvements in performance of token classification systems.
- Sometimes, random words like 'go' and 'on' are selected to be toxic, which means that these types of prepositions and verbs can be removed by exact matching in the string, unless they form parts of larger spans.
- The best checkpoints of the span-based models tend to predict empty spans for the selected example. However, when using checkpoint ensembling, we see that union models return accurate spans.
- The ground spans are not entirely correct and are ambiguous. For example, it is not clear whether the word 'ignorant' should be considered to be toxic. The models, based on other examples, predict 'ignorant' to be toxic, but it is not present in the ground spans. This means that finding the toxic spans is not a trivial task for humans, and annotation can not be performed easily by crowd-workers.
- In some cases, one of the occurrences of the word 'ignorant' is considered to be toxic, while the other is predicted to be benign. The first instance of 'ignorant' does not seem to be as toxic as the second instance and therefore,

more analysis needs to be done to determine the 'degree' of toxicity of the spans. This can be a good direction for future research.

# UoB at SemEval-2021 Task 5: Extending Pre-Trained Language Models to Include Task and Domain-Specific Information for Toxic Span Prediction

**Erik Yan**

School of Computer Science  
University of Birmingham  
United Kingdom

erikdyan@outlook.com

**Harish Tayyar Madabushi**

School of Computer Science  
University of Birmingham  
United Kingdom

Harish@HarishTayyarMadabushi.com

## Abstract

Toxicity is pervasive in social media and poses a major threat to the health of online communities. The recent introduction of pre-trained language models, which have achieved state-of-the-art results in many NLP tasks, has transformed the way in which we approach natural language processing. However, the inherent nature of pre-training means that they are unlikely to capture task-specific statistical information or learn domain-specific knowledge. Additionally, most implementations of these models typically do not employ conditional random fields, a method for simultaneous token classification. We show that these modifications can improve model performance on the Toxic Spans Detection task at SemEval-2021 to achieve a score within 4 percentage points of the top performing team.

## 1 Introduction and Motivation

Moderation is crucial to promoting healthy online discussions. The anonymity afforded by computer-mediated communication enables individuals to engage in toxic behaviour which they would otherwise not consider. Although many datasets and models focusing on toxicity detection have been released, most of them classify entire sequences of text, and do not highlight the individual words that make a text toxic. The Toxic Spans Detection task at SemEval-2021 (Pavlopoulos et al., 2021) focuses on the evaluation of systems that can accurately identify toxic spans within text. Highlighting such spans can provide more information to human moderators in the form of attribution, instead of an unexplained toxicity score per post, and is thus a crucial step towards successful semi-automated moderation. In this paper we focus on the shared task, wherein systems are expected to extract a list of toxic spans, or an empty list, per text. A toxic span is defined as a sequence of words that contributes to a text’s toxicity.

Since 2018, NLP models have adopted the concept of generative pre-training on a diverse corpus of unlabelled text, followed by supervised fine-tuning on specific tasks (Radford et al., 2018). Pre-trained models are built to simulate anthropomorphic learning, wherein existing knowledge can be adapted to new tasks without the need to train on these tasks from scratch - a requirement of traditional machine learning models. This idea of transfer learning, whilst powerful, leads to models being fine-tuned on target tasks using significantly fewer epochs than was previously standard. This reduced training on the target task means that task-specific statistical information or domain-specific knowledge may not be learned by these models.

Such task-specific data may include count-based information, which has been shown to improve the performance of pre-trained models in sequence classification tasks (Lim and Madabushi, 2020; Prakash and Madabushi, 2020), or domain-specific knowledge, such as information pertaining to word toxicity, which has been shown to be one of the most predictive features of offensive commentary (Noever, 2018).

Additionally, pre-trained models tend to use a fully connected layer for classification tasks. This classification layer, however, makes an individual localised prediction for each token without accounting for predictions made on other tokens. A CRF (Lafferty et al., 2001), on the other hand, maximises the probability of the entire sequence of predictions. This makes it more effective for cases where neighbouring predictions may influence each other. NER is one such application, where the decision to assign a certain label to a token may be influenced by the labels assigned to neighbouring tokens. Souza et al. (2020) combined the transfer capabilities of BERT (Devlin et al., 2019) with the structured predictions of a CRF, with the addition of a CRF yielding performance improvements in

several token-level tasks.

Thus, this work aims to test the following hypotheses:

**Hypothesis 1** *Count-based information can aid pre-trained models in token classification tasks.*

**Hypothesis 2** *Pre-trained models are unlikely to capture domain-specific information. Such information is likely to improve their performance in token classification tasks.*

**Hypothesis 3** *Adding a CRF, which affords a sentence-level predictive scope, will improve pre-trained model performance in token classification tasks.*

To ensure reproducibility, our program code, including hyperparameters, is made available online<sup>1</sup>.

## 2 Related Work

Count-based information has been shown to improve the performance of pre-trained models in sequence classification tasks. [Lim and Madabushi \(2020\)](#) proposed an ensemble model of BERT and TF-IDF, which combined the sentence-level information captured by BERT with the corpus-level information provided by TF-IDF. The ensemble model performed 5 percentage points better than a standard BERT model on Subtask A at OffensEval-2020 ([Zampieri et al., 2020](#)), achieving a score within 2 percentage points of the top performing team. Similarly, [Prakash and Madabushi \(2020\)](#) employed an ensemble model of RoBERTa ([Liu et al., 2019b](#)) with a multilayer perceptron using TF-IDF features as input. The ensemble model improved upon the base RoBERTa model by 7 percentage points to achieve state-of-the-art results on the RumourEval-2019 dataset ([Gorrell et al., 2019](#)). We use these studies as a basis for our first hypothesis described in Section 1, and employ a similar method for incorporating TF-IDF features described in Section 3.

Domain-specific information has been shown to be an effective measure of toxicity. [Noever \(2018\)](#) evaluated the relative predictive value of 28 features of syntax, sentiment, emotion, and outlier word dictionaries for online toxicity detection. By rank-ordering features through feature selection, the most predictive feature of offensive commentary was shown to be a simple bad word list. [Peder-](#)

<sup>1</sup>[https://github.com/erikdyan/toxic\\_span\\_detection](https://github.com/erikdyan/toxic_span_detection)

[sen \(2019\)](#) compared two logistic regression classifiers against a simple word list model on Subtask A at OffensEval ([Zampieri et al., 2019](#)), with the rule-based model performing 4 percentage points better than either logistic regression model. Section 3 discusses our methodology for adding word list features, which capture key word information in the offensive language domain, to pre-trained models.

As discussed in Section 1, a CRF is a method for simultaneous token classification which is not commonly employed by pre-trained models. [Souza et al. \(2020\)](#) proposed a BERT-CRF model architecture composed of a token-level classifier on top of a BERT model followed by a linear-chain CRF. Models with a CRF improved upon or performed similarly to models without one on NER tasks in the Portuguese language. This study is, to the best of our knowledge, one of the few that directly compares the performance of a base BERT model against a BERT-CRF model. The improvements arising from adding a CRF supports our third hypothesis described in Section 1, which aims to explore whether similar improvements will arise in the context of toxic span detection.

Submissions to past toxicity detection tasks at SemEval, such as OffensEval and OffensEval-2020, highlight how effective BERT can be for toxicity detection. [Liu et al. \(2019a\)](#) used a fine-tuned BERT model to achieve state-of-the-art results on Subtask A at OffensEval, and seven of the top ten teams used BERT. Similarly, the top ten teams on Subtask A at OffensEval-2020 all used BERT, RoBERTa, or XLM-RoBERTa ([Conneau et al., 2020](#)), sometimes as part of ensemble models with CNNs and LSTMs. ([Wiedemann et al., 2020](#)) submitted the best performing model, which used an ensemble of ALBERT ([Lan et al., 2020](#)) models of different sizes. The success of these models in toxicity detection tasks led us to choose to use a BERT-based model for this work.

## 3 Methodology

Our pre-trained model of choice was DistilBERT ([Sanh et al., 2020](#)), which we used as a baseline measure of performance. We explore and present four models in addition to the baseline DistilBERT model in this paper:



1. DistilBERT+TF-IDF
2. DistilBERT+Word List
3. DistilBERT+TF-IDF+Word List
4. DistilBERT+CRF

To build a basis for comparison, all models were trained using the training data provided by the task organisers and evaluated against the validation dataset. The best performing models were then submitted for evaluation against the test dataset during the task evaluation period. The training process was performed five times, using a different random seed each time. This is because varying the random seed used in fine-tuning BERT models can yield substantially different results, even if the models are the same and identical hyperparameters are used (Dodge et al., 2020). The best performing version of each model was used for the remainder of the study.

In Section 1, we hypothesised that adding count-based information to pre-trained models would improve model performance in token classification tasks. TF-IDF is a count-based statistical measure that captures corpus-level information, accounting for global correlations and associations between words. Use of TF-IDF captures word importance, enabling the identification of key words. This word importance could contribute to the identification of a text’s toxicity, as shown by Lim and Madabushi (2020); Prakash and Madabushi (2020). Thus, we tested our first hypothesis by integrating TF-IDF with the DistilBERT model.

One of the most straightforward approaches for toxicity detection is to use a word list, whereby the toxicity of a sequence is determined by comparing the words it contains against a list of known toxic words. Such domain-specific information has been shown to be effective for toxicity detection (Noever, 2018; Pedersen, 2019). We tested our second hypothesis by adapting a word list feature for token classification and integrating it with the DistilBERT model.

We incorporated the TF-IDF and word list features by modifying the DistilBERT model. First, we removed the token classification layer on top of the baseline DistilBERT model. Then, for the TF-IDF feature, we appended each token’s TF-IDF weight to its hidden state output vector. For the word list feature, we appended a value of 0 or 1. A value of 1 was used if the token appeared in the word list, whilst a value of 0 was used if it did not. These vectors were then pushed through a fully

connected layer for classification.

We tested our third hypothesis by adding a CRF, a method for simultaneous token classification, to the DistilBERT model. We followed the more successful fine-tuning approach used by Souza et al. (2020), which uses a linear classification layer and updates all weights, including BERT’s, during training. The CRF takes the output scores from the classification layer as input and computes the log-likelihood of the given sequence of tags. The model was trained to maximise the log-likelihood of the correct tag sequence.

## 4 Results

Table 1 shows how the best performing version of each model performed when tested against the validation dataset.

Model	F1 Score
DistilBERT	0.58896
DistilBERT+TF-IDF	0.58930
DistilBERT+Word List	0.59296
DistilBERT+TF-IDF+Word List	0.58613
DistilBERT+CRF	0.58615

Table 1: Best F1 score achieved by each model, tested against the validation dataset.

We observe that the inclusion of count-based features did improve model performance, though the increase was very slight. A larger improvement resulted from the use of a word list, whilst model performance worsened when both TF-IDF and word list features were used together and when a CRF was added.

As model performance on the validation dataset was very similar, all models were submitted to the task evaluation stage. Table 2 shows the results of each model tested against the test dataset, whilst Table 3 shows how our best performing model ranked out of the 91 participating teams.

Model	F1 Score
DistilBERT	0.66937
<b>DistilBERT+TF-IDF</b>	<b>0.67609</b>
DistilBERT+Word List	0.67136
DistilBERT+TF-IDF+Word List	0.67393
DistilBERT+CRF	0.67409

Table 2: F1 score achieved by each model, tested against the test dataset.

Rank	Team	F1 Score
1	HITSZ-HLT	0.70830
2	S-NLP	0.70770
3	hitmi&t	0.69848
	...	
<b>25</b>	<b>UoB</b>	<b>0.67609</b>

Table 3: Ranking and F1 score achieved by each team’s best performing model, tested against the test dataset.

It is clear that the F1 scores achieved by all models were very similar when tested against the test dataset, with the DistilBERT+TF-IDF model improving upon the baseline DistilBERT model the most. It is worth noting, however, that whilst this difference is only 0.00672, that same difference would have increased our ranking by 6 ranks had it been added to our final F1 score. Section 5 analyses the significance of the results achieved and studies the differences between the predictions of the DistilBERT and DistilBERT+TF-IDF models in greater detail.

## 5 Discussion and Analysis

It is difficult to conclude with any certainty whether the addition of our proposed features improved model performance, as the scores achieved are very similar. Whilst the increase in performance observed may indeed be due to our additions to the model, we also propose two alternative theories.

The similarity in results may be due to the relative length of the token vectors compared to the length of the additional features. The hidden output from DistilBERT represents each token as a vector of length 768; the addition of one or two elements to each token vector may not be significant enough to discernibly impact model predictions. That being said, there are still small variations in performance between the models. Whilst this may be due to the addition of new features, it may also be due to variations in the random seed used during fine-tuning. Our most improved model performed 0.00672 F1 points better than the baseline DistilBERT model - a figure within the performance variation range observed during tests involving the random seed (the DistilBERT+TF-IDF model ranged by 0.00922 from 0.58008 to 0.58930, for example). Despite our efforts to counteract this, time and resource limitations meant we were only able to train each model five times instead of the more rigorous ten.

We conduct a more detailed analysis into the

predictions of the baseline DistilBERT model and our best performing (DistilBERT+TF-IDF) model. Tables 4 and 5 show the confusion matrix of each model, respectively. These matrices are constructed using a subset of the test dataset from which the tokens correctly predicted by both models to be non-toxic have been removed. We subset the dataset in this way to significantly reduce the size of the data and to remove less interesting tokens. Table 4 shows that the baseline DistilBERT model tends to overpredict toxic labels, resulting in 1466 false positives. Table 5 shows that the addition of the TF-IDF feature helps to mitigate this, with the DistilBERT+TF-IDF model correctly predicting over 100 of DistilBERT’s false positives as true negatives - an overall improvement of 2.5% on this subset of the test set.

		Predicted	
		Non-Toxic	Toxic
True	Non-Toxic	202	1466
	Toxic	682	1829

Table 4: Confusion matrix of the DistilBERT model on a subset of the test dataset from which the tokens correctly predicted by both the DistilBERT and DistilBERT+TF-IDF models to be non-toxic have been removed.

		Predicted	
		Non-Toxic	Toxic
True	Non-Toxic	310	1358
	Toxic	692	1819

Table 5: Confusion matrix of the DistilBERT+TF-IDF model on a subset of the test dataset from which the tokens correctly predicted by both the DistilBERT and DistilBERT+TF-IDF models to be non-toxic have been removed.

In addition to an exploration of the results and confusion matrix, we perform an error analysis by manually comparing the true labels and predictions of the DistilBERT and DistilBERT+TF-IDF models. We first observe that there are some inconsistencies in the true labels. For example, the phrase “... racist, sexist, narcissistic, pathological liar ...” is marked as non-toxic, whereas “... sexist rubbish ...” is marked as toxic. Another trend observed is that both models struggle to correctly predict phrases containing ordinarily non-toxic words which become toxic given the context. For example, consider the phrases “Trump troll”, “Bunch of

cowards”, “Total rubbish”, and “PATHETIC LIB LOSER”. Whilst the true labels classify all of these phrases as toxic, both models only predicted the words “troll”, “cowards”, “rubbish”, “pathetic”, and “loser” to be toxic.

These trends highlight some of the inherent difficulties involved in token classification tasks for both machine learning models and human annotators.

## 6 Conclusion and Future Work

This work explored the possibility of improving pre-trained model performance on the token classification task of toxic span detection. As discussed in Section 1, we hypothesised that adding 1) count-based information, 2) domain-specific knowledge, and 3) a CRF can aid pre-trained models in token classification tasks. Whilst our experimental results (Section 4) seem to suggest that all three of these features improve the performance of DistilBERT, we note that they do so only marginally (Section 5). Further analysis, however, showed that, whilst the overall F1 improvement from adding TF-IDF was small, the addition of a count-based feature helped to reduce DistilBERT’s overprediction of toxic tokens.

We believe that these improvements, whilst small, provide an interesting avenue of exploration. We intend to further explore how these and other similar features interact with pre-trained models in the task of token classification.

## References

- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. [Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping](#).
- Genevieve Gorrell, Elena Kochkina, Maria Liakata, Ahmet Aker, Arkaitz Zubiaga, Kalina Bontcheva, and Leon Derczynski. 2019. [SemEval-2019 task 7: RumourEval, determining rumour veracity and support for rumours](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 845–854, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- John Lafferty, Andrew McCallum, and Fernando C.N. Pereira. 2001. [Conditional random fields: Probabilistic models for segmenting and labeling sequence data](#).
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [Albert: A lite bert for self-supervised learning of language representations](#).
- Wah Meng Lim and Harish Tayyar Madabushi. 2020. [Uob at semeval-2020 task 12: Boosting bert with corpus level information](#).
- Ping Liu, Wen Li, and Liang Zou. 2019a. [NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Roberta: A robustly optimized bert pretraining approach](#).
- David Noever. 2018. [Machine learning suites for online toxicity detection](#).
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Ted Pedersen. 2019. [Duluth at SemEval-2019 task 6: Lexical approaches to identify and categorize offensive tweets](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 593–599, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Anushka Prakash and Harish Tayyar Madabushi. 2020. [Incorporating count-based features into pre-trained models for improved stance detection](#).
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. [Improving language understanding by generative pre-training](#).
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).
- Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020. [Portuguese named entity recognition using bert-crf](#).

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. Uhh-It at semeval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval).

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020).

# Cisco at SemEval-2021 Task 5: What’s Toxic?: Leveraging Transformers for Multiple Toxic Span Extraction from Online Comments

**Sreyan Ghosh**

Cisco Systems, Bangalore, India  
MIDAS Lab, IIT-Delhi, India  
sreyghos@cisco.com

**Sonal Kumar**

Cisco Systems, Bangalore, India  
sonalkum@cisco.com

## Abstract

Social network platforms are generally used to share positive, constructive, and insightful content. However, in recent times, people often get exposed to objectionable content like threat, identity attacks, hate speech, insults, obscene texts, offensive remarks or bullying. Existing work on toxic speech detection focuses on binary classification or on differentiating toxic speech among a small set of categories. This paper describes the system proposed by team Cisco for SemEval-2021 Task 5: Toxic Spans Detection, the first shared task focusing on detecting the spans in the text that attribute to its toxicity, in English language. We approach this problem primarily in two ways: a sequence tagging approach and a dependency parsing approach. In our sequence tagging approach we tag each token in a sentence under a particular tagging scheme. Our best performing architecture in this approach also proved to be our best performing architecture overall with an  $F_1$  score of **0.6922**, thereby placing us 7<sup>th</sup> on the final evaluation phase leaderboard. We also explore a dependency parsing approach where we extract spans from the input sentence under the supervision of target span boundaries and rank our spans using a biaffine model. Finally, we also provide a detailed analysis of our results and model performance in our paper.

## 1 Introduction

It only takes one toxic comment to sour an online discussion. The threat of abuse and harassment online leads many people to stop expressing themselves and give up on seeking different opinions. Toxic content is ubiquitous in social media platforms like Twitter, Facebook, Reddit, the increase of which is a major cultural threat and has already lead to a crime against minorities (Williams et al., 2020). Toxic text in online social media varies depending on targeted groups (e.g. women, LGBT,

gay, African, immigrants) or the context (e.g. pro-trump discussion or the metoo movement). Toxic Text online has often been broadly classified by researchers into different categories like hate, offense, hostility, aggression, identity attacks, and cyberbullying. Though the use of various terms for equivalent tasks makes them incomparable at times (Fortuna et al., 2020), toxic speech or spans in this particular task, SemEval-2021 Task 5 (Pavlopoulos et al., 2021), has been considered as a super-set of all the above sub-types.

1. Two new contestants for America's **Dumbest Criminals**.
2. **Stealing** is what you do best.

Figure 1: Toxic spans in sentences

While a lot of models have claimed to achieve state-of-the-art results on various datasets, it has been observed that most models fail to generalize (Arango et al., 2019; Gröndahl et al., 2018). The models tend to classify comments as toxic that have a reference to certain commonly-attacked entities (e.g. gay, black, Muslim, immigrants) without the comment having any intention to be toxic (Dixon et al., 2018; Borkan et al., 2019). A large vocabulary of certain trigger terms leads to a biased prediction by the models (Sap et al., 2019; Davidson et al., 2017). Thus, it has become increasingly important in recent times to determine parts of the text that attribute to the toxic nature of the sentence, for both automated and semi-automated content moderation on social media platforms, primarily for the purpose of helping human moderators deal with lengthy comments and also provide them attributions for better explainability on the toxic nature of the post. This in turn would aid in better handling of unintended bias in toxic text classification. SemEval-2021 Task 5: Toxic Spans Detection focuses on exactly this problem of detecting toxic

spans from sentences already classified as toxic on a post-level.

In this paper, we approach the problem of multiple non-contiguous toxic span extraction from texts both as a *sequence tagging task* and as a standard *span extraction task* resembling the generic approach and architecture adopted for single-span Reading Comprehension (RC) task. For our sequence tagging approach, we predict for each token, whether it is a part of the span. For our second approach, we predict and compute a couple of scores for each token, corresponding to whether that token is the start or end of the span. In addition to this, we deploy a biaffine model to score start and end indices, thus adopting the methodology for multiple non-contiguous span extraction.

## 2 Literature

Previous work on automated toxic text detection, and its various sub-types, focuses on developing classifiers that can flag toxic content with a high degree of accuracy on datasets curated from various social media platforms in English (Carta et al., 2019; Saeed et al., 2018; Vaidya et al., 2020), other foreign languages (Zhang et al., 2018; Mishra et al., 2018; Qian et al., 2019; Davidson et al., 2017; Kamal et al., 2021; Leite et al., 2020) including code-switched text (Mathur et al., 2018a,b; Kapoor et al., 2019) and multilingual text (Zampieri et al., 2019). This topic has also evidenced a number of workshops (Kumar et al., 2018) and competitions (Zampieri et al., 2019, 2020; Basile et al., 2019; Mandl et al., 2019).

Recent work shows transformer based architectures like BERT (Devlin et al., 2019) have been performing well on the task of offensive language classification (Liu et al., 2019a; Safaya et al., 2020; Dai et al., 2020). Transformer based architectures have also produced state-of-the-art performance on sequence tagging tasks like *Named Entity Recognition (NER)* (Yamada et al., 2020; Devlin et al., 2019; Yang et al., 2019) *span extraction* (Eberts and Ulges, 2019; Joshi et al., 2020) and *QA tasks* (Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020). Multiple span extraction from texts has been explored both as a *sequence tagging task* (Patil et al., 2020; Segal et al., 2019) and as span extraction as in RC tasks (Hu et al., 2019; Yu et al., 2020).

Very recently HateXplain (Mathew et al., 2020) proposed a benchmark dataset for explainable hate speech detection using the concept of rationales.

Attempts have also been made to handle identity bias in toxic text classification (Vaidya et al., 2020) and also to make robust toxic text classifiers which help adversaries not bypass toxic filters (Kurita et al., 2019).

## 3 Methodology

For our sequence tagging approach, we explore two tagging schemes. First, the well known *BIO* tagging scheme, where *B* indicates the first token of an output span, *I* indicates the subsequent tokens and *O* denotes the tokens that are not part of the output span. Additionally, we also try a simpler *IO* tagging scheme, where words which are part of a span are tagged as *I* or *O* otherwise. Formally, given an input sentence  $\mathbf{x} = (x_1, \dots, x_n)$ , of length  $n$ , and a tagging scheme with  $|S|$  tags ( $|S| = 3$  for *BIO* and  $|S| = 2$  for *IO*), for each of  $n$  tokens the probability for the tag of the  $i$ -th token is

$$\mathbf{p}_i = \text{softmax}(f(\mathbf{h}_i)) \quad (1)$$

where  $\mathbf{p} \in R^{m \times |S|}$ , and  $f$  is parameterized function with  $|S|$  outputs.

Our other approach is based on the standard single-span extraction architecture widely used for RC Tasks. With this approach, we extract toxic spans from sentences under the supervision of target span boundaries, but with an added biaffine model for scoring the multiple toxic spans instead of simply taking top  $k$  spans based on the start and end probabilities, thus giving our model a global view of the input. The main advantage of this approach is that the extractive search space can be reduced linearly with the sentence length, which is far less than the sequence tagging method. Given an input sentence  $\mathbf{x} = (x_1, \dots, x_n)$ , of length  $n$ , we predict a target list  $\mathbf{T} = (t_1, \dots, t_m)$  where the number of targets is  $m$  and each target  $t_i$  is annotated with its start position  $s_i$ , its end position  $e_i$  and the class that span belongs to (only one in our case, *toxic*).

However, to adapt to the problem of extracting multiple spans from the sentence, instead of taking the top  $k$  spans based on the start and end probabilities, we apply a biaffine model (Dozat and Manning, 2016) to score all the spans with the constraint  $s_i \leq e_i$ . Post this we rank all the spans in descending order and choose every span as long it does not clash with higher-ranked spans.

## 4 Dataset

The dataset provided to us by the organizers of the workshop consisted of a random subset of 10,000 posts from the publicly available Civil Comments Dataset, from a set of 30,000 posts originally annotated as toxic (or severely toxic) on post-level annotations, manually annotated by 3 crowd-raters per post for toxic spans. The final character offsets were obtained by retaining the offsets with a probability of more than 50%, computed as a fraction of raters who annotated the character offsets as toxic. Basic statistics about the dataset can be found in Table 1.

	Sentences	Spans
Train	7939	10298
Dev	690	903
Test	2000	1850

Table 1: Number of sentences and spans

Additionally, we provide a quick look into the length-wise distribution of spans across the train, development, and test set in Table 2. As we observe, the majority of the spans are just a single word in length and mostly comprise of the most commonly used cuss words in the *English* language. In our Results Analysis section, we show how this metric stands important for training and evaluating our systems and for the future development of toxic span extraction datasets.

	Train	Dev	Test
1	7897	687	1650
2-4	1617	153	174
$\geq 5$	784	63	26

Table 2: Length-wise segregation of the number of non-contiguous spans

## 5 Evaluation Metric

To evaluate the performance of our systems we employ  $F_I$  as used by [Da San Martino et al. \(2019\)](#). Let system  $A$  return a set  $S_A^t$  of character offsets, for parts of the post found to be toxic. Let  $S_G^t$  be the character offsets of the ground truth annotations of post  $t$ . We calculate  $F_I$  score of  $S_A^t$  w.r.t  $S_G^t$  as follows where  $|\cdot|$  denotes set cardinality.

$$P^t(A, G) = \frac{|S_A^t \cap S_G^t|}{|S_A^t|} \quad (2)$$

$$R^t(A, G) = \frac{|S_A^t \cap S_G^t|}{|S_G^t|} \quad (3)$$

$$F_1^t(A, G) = \frac{2 \cdot P^t(A, G) \cdot R^t(A, G)}{P^t(A, G) + R^t(A, G)} \quad (4)$$

If predicted span i.e  $S_A^t$  is empty for a post  $t$  then we set  $F_1^t(A, G) = 1$  if the gold truth i.e  $S_G^t$  is also empty, else if  $S_G^t$  is empty and  $S_A^t$  is not empty then we set  $F_1^t(A, G) = 0$ .

## 6 System Description

### 6.1 Sequence Tagging Approach

For our sequence tagging approach we employ the commonly used BiLSTM-CRF architecture ([Huang et al., 2015](#)) used predominately in many sequence tagging problems, but with added contextual word embeddings for each word using transformer and character-based word embeddings. We experiment with a total of 5 transformer architectures, namely BERT ([Devlin et al., 2019](#)), XLNet ([Yang et al., 2019](#)), RoBERTa ([Liu et al., 2019b](#)), ALBERT ([Lan et al., 2020](#)) and SpanBERT ([Joshi et al., 2020](#)). For all of the above mentioned transformer architectures, the *large* variant of the transformer was used except ALBERT for which we use its *xlarge-v2* variant. First, the tokenized word input is passed through the transformer architecture and the output of the last 4 encoder layers is concatenated to obtain the final contextualized word embedding  $E_T$  for each word in the sentence. Additionally, we also pass each character in a word through a character-level BiLSTM network, to obtain character-based word embeddings for the word  $E_C$  as used by [Lample et al. \(2016\)](#). Finally, both these word embeddings,  $E_T$  and  $E_C$ , for each word are concatenated and passed through a BiLSTM layer followed by a CRF layer to obtain the best probable tag for each word in the sentence.

### 6.2 Dependency Parsing Approach

For our dependency parsing approach, we employ a similar approach as proposed by [Yu et al. \(2020\)](#), using a biaffine classifier to score our spans post-extraction. This methodology fits best to our purpose of *multiple* toxic span extraction from sentences compared to span extraction systems in general RC tasks which are capable of extracting just a single span from a sentence ([Yang and Ishfaq](#)). For each word first we extract it’s BERT, FasText

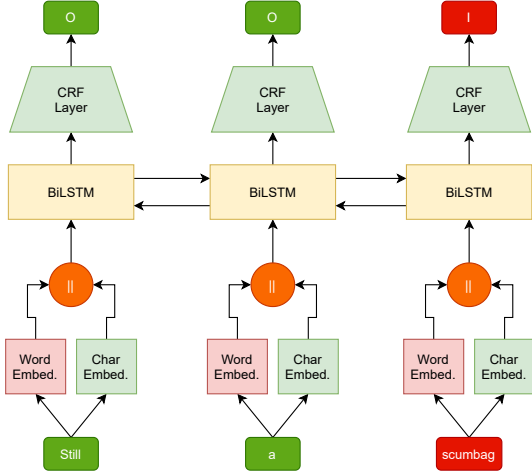


Figure 2: Sequence Tagger Model

and character-based word embeddings. We used  $BERT_{Large}$  for all our experiments and used the recipe followed by Kantor and Globerson (2019) to extract contextual embeddings for each token. After concatenating both the word embeddings and character embeddings for each word, we feed the output to a BiLSTM layer. We then apply two separate FFNNs to the output word representations  $x$  to create different representations ( $h_s / h_e$ ) for the start/end of the spans. These representations are then passed through a biaffine model for scoring all possible spans  $(s_i, e_i)$ , where  $s_i$  and  $e_i$  are start and end indices of the span, under the constraint  $s_i \leq e_i$  (the start of the span is before its end) by creating a  $l \times l \times c$  scoring tensor  $r_m$ , where  $l$  is the length of the sentence and  $c$  is the number of NER categories + 1 (for non-entity). We compute the score for a span  $i$  by:

$$h_s(i) = FFNN_s(x_{s_i}) \quad (5)$$

$$h_e(i) = FFNN_e(x_{e_i}) \quad (6)$$

$$r_m(i) = h_s(i)^T U_m h_e(i) + W_m (h_s(i) \oplus h_e(i)) + b_m \quad (7)$$

We finally assign each span a category  $y'$  based on

$$y'(i) = \arg \max r_m(i) \quad (8)$$

Post this, we rank each span that has a category other than non-entity and consider all the spans for our final prediction as long as it does not clash with higher ranked spans with an additional constraint,

whereby, an entity containing or is inside an entity ranked before it will not be selected.

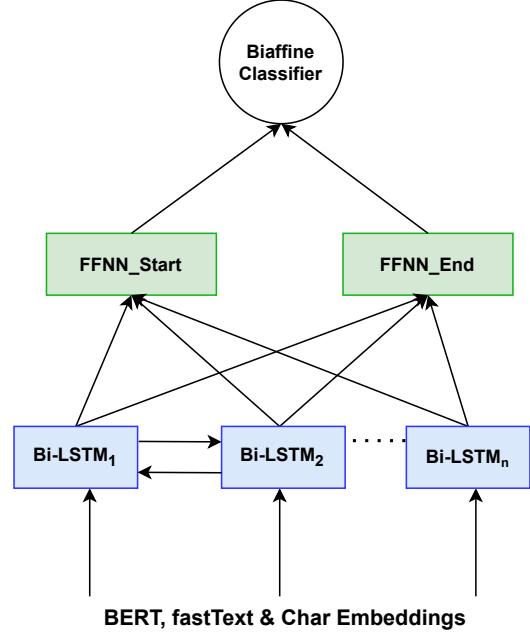


Figure 3: Biaffine Model

## 7 Experimental Setup

Data was originally provided to us in the form of sentences and the corresponding character offsets for the toxic spans of the sentence. Before converting the character offsets to our required format for our respective approaches, we apply some basic text pre-processing to all our sentences. First, we normalize all the sentences by converting all white-space characters to spaces. Second, we split all punctuation characters from both sides of a word and also break abbreviated words. These pre-processing steps help improve the  $F_1$  score of both our approaches as shown in Table 6. Post these pre-processing steps, we formulate our targets for both our approaches. For our sequence tagging approach, we tag each word in the sentence with its corresponding tag based on the tagging scheme we follow, *BIO* or *IO*. For our span extraction approach, we convert the sequence of character offsets into its corresponding word-level start and end indices for each span. In Fig. 4, we provide a pictorial representation of the above mentioned procedures we follow for data preparation for both our approaches.

We use PyTorch<sup>1</sup> Framework for building our Deep Learning models along with the Transformer

<sup>1</sup><https://pytorch.org/>



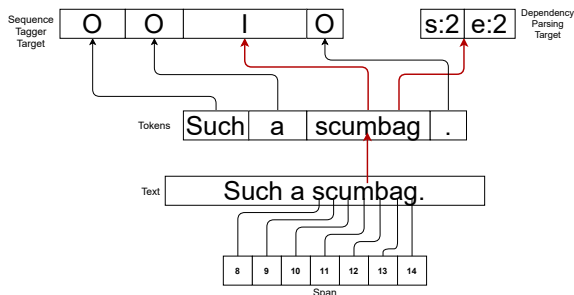


Figure 4: Data Preparation

implementations, pre-trained models and, specific tokenizers in the HuggingFace<sup>2</sup> library.

We mention the major hyperparameters of our best-performing systems experimental setting for our dependency parsing approach and span extraction approach in Tables 3 and 4 respectively.

Parameter	Value
BiLSTM size	256
BiLSTM layer	1
BiLSTM dropout	0
Transformer size	1024
Transformer encoder layers	last 4
Char BiLSTM Hidden Size	25
Char BiLSTM layers	1
Optimiser	Adam
Learning rate	[1e-3,1.56e-4]

Table 3: Major hyperparameters of Sequence Tagger model

Parameter	Value
BiLSTM size	200
BiLSTM layer	3
BiLSTM dropout	0.4
FFNN size	150
FFNN dropout	0.2
BERT size	1024
BERT encoder layers	last 4
fastText embedding size	300
Char CNN size	50
Char CNN filter width	[3,4,5]
Embeddings dropout	0.5
Optimiser	Adam
Learning rate	1e-3

Table 4: Major hyperparameters of Dependency Parsing model

We train all our sequence tagging models with

<sup>2</sup><http://huggingface.co/>

stochastic gradient descent in batched mode with a batch size of 8. In the training phase, we keep all layers in our model, including all the transformer layers trainable. We start training our model at a learning rate of 0.01, with a minimum threshold limit of 0.0001, and half the learning rate after every 4 consecutive epochs of no improvement in the  $F_1$  score of the development set. We train our model to a maximum of 100 epochs or 4 consecutive epochs of no improvement at our minimum learning rate.

We train our our model for dependency parsing approach with Adam optimizer in batched mode with a batch size of 32 and a learning rate of 0.0001 for a maximum of 40,000 steps. With this approach too, we keep all layers trainable in the training phase except the BERT Transformer layers. Pre-trained BERT and fastText embeddings were just used to extract context-dependent and independent embeddings respectively and BERT was *not fine-tuned* in the training phase.

The training was performed on 1 NVIDIA Titan X GPU. Our code is available on Github<sup>3</sup>.

## 8 Results

In Table 5 we present  $F_1$  scores for all our systems trained for both our sequence tagging and span extraction approaches. For our sequence tagging approach, we divide our results according to the transformer architecture and tagging scheme used for that experiment.

Model	Scheme	Test	Dev
XLNet	IO	<b>0.6922</b>	0.6945
XLNet	BIO	0.6653	0.6683
spanBERT	IO	0.6777	0.6744
spanBERT	BIO	0.6887	0.6730
RoBERTa	IO	0.6647	<b>0.6967</b>
RoBERTa	BIO	0.6849	0.6789
BERT	IO	0.6830	0.6814
BERT	BIO	0.6852	0.6815
ALBERT	IO	0.6621	0.6702
ALBERT	BIO	0.6679	0.6431
Biaffine	-	0.6731	0.6627

Table 5: Test and Dev Results of different models on various tagging scheme

Our best performing architecture proved to be the sequence tagging system with XLnet trans-

<sup>3</sup><https://github.com/Sreyan88/SemEval-2021-Toxic-Spans-Detection>

former trained with *IO* tagging scheme. Additionally, in Table 6 we show how the LSTM and CRF over the transformer architecture, and our pre-processing step mentioned in Section 7 affect the performance of our best performing architecture.

	F1	$\Delta$
Our Model	0.6922	-
- LSTM	0.6912	0.0010
- CRF	0.6850	0.0072
- Pre-processing	0.6759	0.1630

Table 6: Impact of LSTM, CRF and pre-processing on learning

## 9 Results Analysis

### 9.1 Length vs Performance

We wanted to understand how the performance of the system varied with varying lengths of spans. Table 7 summarizes the performance of our best performing systems on all approaches experimented by us, on the test dataset spans, divided into 3 sets according to their length in terms of the number of words that help to make the span.

Model	Span length	F1
Seq. Tagger (IO)	1	<b>0.6546</b>
	2-4	0.1750
	$\geq 5$	0.0596
Seq. Tagger (BIO)	1	<b>0.6588</b>
	2-4	0.1524
	$\geq 5$	0.09198
Dependency Parsing	1	<b>0.6486</b>
	2-4	0.0514
	$\geq 5$	0.0

Table 7: Span Length vs. Performance

### 9.2 Learning context

Majority of single word spans in the dataset are the most commonly used cuss words or abusive words in the English language, i.e., words that can be directly classified as toxic and are not context-dependant, e.g. "stupid", "idiot" etc., with spans longer than a single word having a lesser ratio of such words. We acknowledge the fact that an AI-based system should be able to do much more, like learning the context behind which a word is used, than just detect common *English* cuss words from a sentence, which can be otherwise done by a simple

1. So you agree that **black children should be killed**. I got it. So much for innocent until proven guilty. That is a white privilege too.
2. Good luck with those emerging markets. Your state is over saturated with commercial grows and the price is plummeting. He will have to put his stuff on the **black** market and that won't work either. He's an **idiot**.

Figure 5: Toxicity classification of the word "black" in toxic and non-toxic context

dictionary search. The deteriorating performance of the model with an increase in span length makes us dig deeper into our test set results to find out if our model is being able to detect *context-based* toxic spans from sentences. We follow a two step procedure to analyze this. First, we calculate our model performance on single-word spans consisting of just the top 25 most commonly occurring context-independent cuss words<sup>4</sup>. Table 8 shows an analysis of these results. Second, we take the word "black" and analyze two sentences in our test where the word black was mentioned in a toxic and non-toxic context. Fig. 5 shows how our model indeed tags the latter black as toxic and the former one as non-toxic.

Single Word Cuss Spans	Others
0.6894	0.1736

Table 8: F<sub>1</sub> score of context independent cuss words

## 10 Conclusion

In this paper, we present our approach to SemEval-2021 Task 5: Toxic Spans Detection. Our best submission gave us an *F<sub>1</sub>* score of **0.6922**, placing us 7<sup>th</sup> on the Evaluation Phase Leaderboard. Future work includes independently incorporating both post level and sentence level context for determining the toxicity of a word, and also collating a dataset with toxic spans comprising of a healthy mixture of simple cuss words (which can always be attributed as toxic independent of the context) and words for which the toxicity of the word depends on the context in which it appears, thereby making better systems towards *contextual* toxic span detection.

<sup>4</sup>List of cuss words used for analysis can be found in our GitHub repository

## References

- Aymé Arango, Jorge Pérez, and Barbara Poblete. 2019. Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–54.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. [SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 491–500.
- Salvatore Carta, Andrea Corriga, Riccardo Mulas, Diego Reforgiato Recupero, and Roberto Saia. 2019. A supervised multi-class multi-label word embeddings approach for toxic comment classification. In *KDIR*, pages 105–112.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.
- Wenliang Dai, Tiezheng Yu, Zihan Liu, and Pascale Fung. 2020. Kungfupanda at semeval-2020 task 12: Bert-based multi-task learning for offensive language detection. *arXiv preprint arXiv:2004.13432*.
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.
- Timothy Dozat and Christopher D. Manning. 2016. [Deep biaffine attention for neural dependency parsing](#). *CoRR*, abs/1611.01734.
- Markus Eberts and Adrian Ulges. 2019. Span-based joint entity and relation extraction with transformer pre-training. *arXiv preprint arXiv:1909.07755*.
- Paula Fortuna, Juan Soler, and Leo Wanner. 2020. Toxic, hateful, offensive or abusive? what are we really classifying? an empirical analysis of hate speech datasets. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 6786–6794.
- Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N Asokan. 2018. All you need is” love” evading hate speech detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 2–12.
- Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019. Open-domain targeted sentiment analysis via span-based extraction and classification. *arXiv preprint arXiv:1906.03820*.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Ojasv Kamal, Adarsh Kumar, and Tejas Vaidhya. 2021. [Hostility detection in hindi leveraging pre-trained language models](#).
- Ben Kantor and Amir Globerson. 2019. [Coreference resolution with entity equalization](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 673–677, Florence, Italy. Association for Computational Linguistics.
- Raghav Kapoor, Yaman Kumar, Kshitij Rajput, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. 2019. Mind your language: Abuse and offense detection for code-switched languages. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9951–9952.
- Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 1–11.
- Keita Kurita, Anna Belova, and Antonios Anastasopoulos. 2019. Towards robust toxic content classification. *arXiv preprint arXiv:1912.06872*.

- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [Albert: A lite bert for self-supervised learning of language representations](#). In *International Conference on Learning Representations*.
- Joao A Leite, Diego F Silva, Kalina Bontcheva, and Carolina Scarton. 2020. Toxic language detection in social media for brazilian portuguese: New dataset and multilingual analysis. *arXiv preprint arXiv:2010.04543*.
- Ping Liu, Wen Li, and Liang Zou. 2019a. Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 87–91.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of the 11th Forum for Information Retrieval Evaluation*, pages 14–17.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.
- Puneet Mathur, Ramit Sawhney, Meghna Ayyar, and Rajiv Shah. 2018a. Did you offend me? classification of offensive tweets in hinglish language. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 138–148.
- Puneet Mathur, Rajiv Shah, Ramit Sawhney, and Debanjan Mahata. 2018b. Detecting offensive tweets in hindi-english code-switched language. In *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, pages 18–26.
- Pushkar Mishra, Marco Del Tredici, Helen Yanakoudakis, and Ekaterina Shutova. 2018. Author profiling for abuse detection. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1088–1098.
- Rajaswa Patil, Somesh Singh, and Swati Agarwal. 2020. Bpgc at semeval-2020 task 11: Propaganda detection in news articles with multi-granularity knowledge sharing and linguistic features based ensemble learning. *arXiv preprint arXiv:2006.00593*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. *arXiv preprint arXiv:1909.04251*.
- Hafiz Hassaan Saeed, Khurram Shahzad, and Faisal Kamiran. 2018. Overlapping toxic sentiment classification using deep neural architectures. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 1361–1366. IEEE.
- Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059.
- Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678.
- Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2019. A simple and effective model for answering multi-span questions. *arXiv preprint arXiv:1909.13375*.
- Ameya Vaidya, Feng Mai, and Yue Ning. 2020. Empirical analysis of multi-task learning for reducing identity bias in toxic comment detection. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 683–693.
- Matthew L Williams, Pete Burnap, Amir Javed, Han Liu, and Sefa Ozalp. 2020. Hate in the machine: anti-black and anti-muslim social media posts as predictors of offline racially and religiously aggravated crime. *The British Journal of Criminology*, 60(1):93–117.
- Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: deep contextualized entity representations with entity-aware self-attention. *arXiv preprint arXiv:2010.01057*.
- Chenjie Yang and Haque Ishfaq. Question answering on squad.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. *arXiv preprint arXiv:2005.07150*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer.

# MedAI at SemEval-2021 Task 5: Start-to-end Tagging Framework for Toxic Spans Detection

Zhen Wang<sup>1</sup>, Hongjie Fan<sup>2</sup>\*, Junfei Liu<sup>3</sup>

School of Software and Microelectronics, Peking University  
The Department of Science and Technology, China University of Political Science and Law  
National Engineering Research Center for Software Engineering, Peking University  
{wang.zh, liujunfei}@pku.edu.cn, hjfan@cupl.edu.cn

## Abstract

This paper describes the system submitted to SemEval 2021 Task 5: Toxic Spans Detection. The task concerns evaluating systems that detect the spans that make a text toxic when detecting such spans are possible. To address the possibly multi-span detection problem, we develop a start-to-end tagging framework on the top of RoBERTa based language model. Besides, we design a custom loss function which take distance into account. In comparison to other participating teams, our system has achieved 69.03% F1 score, which is slight lower (-1.8 and -1.73) than the top 1 (70.83%) and top 2 (70.77%), respectively.

## 1 Introduction

In recent years, social networks and microblogging sites' popularity have increased, attracting more users. With a huge user base, social media will continue to publish a large amount of user-generated content. As the use of social media increased, other undesirable phenomena and behaviors emerged. Social media users often abuse this freedom to spread abusive or hateful posts or comments. In many cases, the user-generated content is offensive or proactive, and users may have to deal with threats such as cyberattacks or cyberbullying, and other undesirable (Warner and Hirschberg 2012). Therefore, the issue of detecting and possibly limiting the spread of toxic post has become increasingly important.

Although several toxicity or abusive language detection datasets (Wulczyn et al. 2016; Borkan et al. 2019) and models (Borkan et al. 2019; Pavlopoulos et al. 2017; Zampieri et al. 2019) have been released, most of them classify whole comments or documents, and do not identify the spans that make a text toxic. But highlighting such toxic spans can

assist human moderators (e.g., news portals moderators) who often deal with lengthy comments, and who prefer attribution instead of just a system-generated unexplained toxicity score per post. The evaluation of systems that could accurately locate toxic spans within a text is thus a crucial step towards successful semi-automated moderation.

For this reason, SemEval 2021 set up the task Toxic Spans Detection to detect and extract the spans that make a text toxic, when detecting such spans is possible (Pavlopoulos et al. 2021). To address the possibly multi-span extraction problem, we develop a start-to-end tagging framework with custom distance loss, which can tag the start and end position of a toxic span. Based on this scheme, we can effectively deal with the multi-span extraction problem.

The rest of the paper is organized as follows: Section 2 provides system overview. Section 3 describes our approach in detail. Our experiment is discussed in Section 4. We conclude our work in Section 5.

## 2 System Overview

### 2.1 Preprocessing and Word Embedding

The training dataset contains 3 columns:

ID - Contains a unique number to identify each training example.

Spans - Contains a list of indexes that indicates the position of toxic spans.

Text - Contains the text that need to detect and extract the toxic spans.

Note that the spans are not given in text, We transformed the indexes to text first. Besides, we append the "negative" word to the end of each post serving as the indicator. We use word embeddings as input to the model. Word embedding is a distributed vector representation of words (Mikolov et al. 2013), capturing the syntactic and semantic in-

\* Corresponding author.

formation of words. Effective word embedding can get better performance. After comparison, we use the RoBERTa-based pre-training language models as sentence encoder for word embedding.

## 2.2 Sequence Tagging

Sequence tagging as a general methods can be used in wide applications, such as named entity recognition, relation extraction, machine reading comprehension and so on.

The tagging scheme can be divided into BIO (Zheng et al. 2017), BIOES (Huang et al. 2015) and others, in which B denotes the first token of an output span, I denotes subsequent tokens in a span, O denotes tokens that are not part of an output span, E denotes the last token of an output span and S denotes token that is an output span.

## 3 Model Description

Our model has two steps as follows: 1. Concatenate the "negative" word at end of each post. 2. Obtain the word embedding of each token in the post to form the final representation and predict the start and end probabilities for each token as output.

Figure 1 shows the general structure of the system. More details for the systems components are shown in the following subsections.

### 3.1 Embedding Layer

As input sequence  $X$  of length  $T$  is composed of word tokens:  $X = \{x_1, \dots, x_T\}$ . Each token  $x_t$  is replaced with the corresponding vocabulary index  $V(t)$ . The embedding layer transforms the token into vector  $e_t \in R^d$  which is selected from the embedding matrix  $E$  according to the index, where  $d$  is the dimensionality of the embedding space.

In order to indicate the model extract toxic or negative spans, we append the word embedding vector of "negative" to the end of each post. We take the mean of last two hidden layer's weight as word embedding. The example of sentence constructed is also shown in Figure 1.

### 3.2 Tagging scheme

Although the classical BIOES tag based model can obtain competitive result, we think the training dataset is not big enough to learn so many tags. So different the above methods, we apply the start-to-end tagging scheme that predicting start and end probabilities for each token. The different target sequence used by several loss function are as shown in Figure 1.

## 3.3 Loss Function

### 3.3.1 Classical Cross-Entropy Loss

At the beginning, we use the classical binary cross-entropy loss, which creates a criterion that measures the Binary Cross Entropy between the target and the output. The loss can be described as:

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top$$

$$l_n = -[y_n \cdot \log x_n + (1 - y_n) \cdot \log (1 - x_n)]$$

### 3.3.2 Label Smoothing Loss

Consider that, we are using roBERTa(Liu et al. 2019) as encoder, which is a large pre-trained language model. and may cause the over-fitting problem. To prevent this, we apply the label smoothing(Szegedy et al. 2015) method and change the '0' in the target sequence to small value 0.025. The computation method is same with cross-entropy loss.

### 3.3.3 Kullback-Leibler Divergence Loss

Besides the handcrafted label smoothing loss, we also tried the KLDivLoss, which is a useful distance measure for continuous distributions and is often useful when performing direct regression over the space of (discretely sampled) continuous output distributions.

The target sequence is the same with the above binary cross-entropy loss. The loss can be described as:

$$l(x, y) = L = \{l_1, \dots, l_N\}$$

$$l_n = y_n \cdot (\log y_n - x_n)$$

where the index  $N$  spans all dimensions of input and  $L$  has the same shape as input.

### 3.3.4 Custom Distance Loss

We notice that the cross-entropy loss pay equal weight to each position's loss, no matter how far the distance between it and the target. To penalize more on the distant false prediction, we propose a custom distance loss, which use an auxiliary sequence that generated by insert equal interval from 0 to 1 center on the '1' target. And use the mean dot product to compute the distance loss.

## 4 Evaluation

### 4.1 Data

The shared task provides trail, training and testing datasets to be used by all participants. The statistics

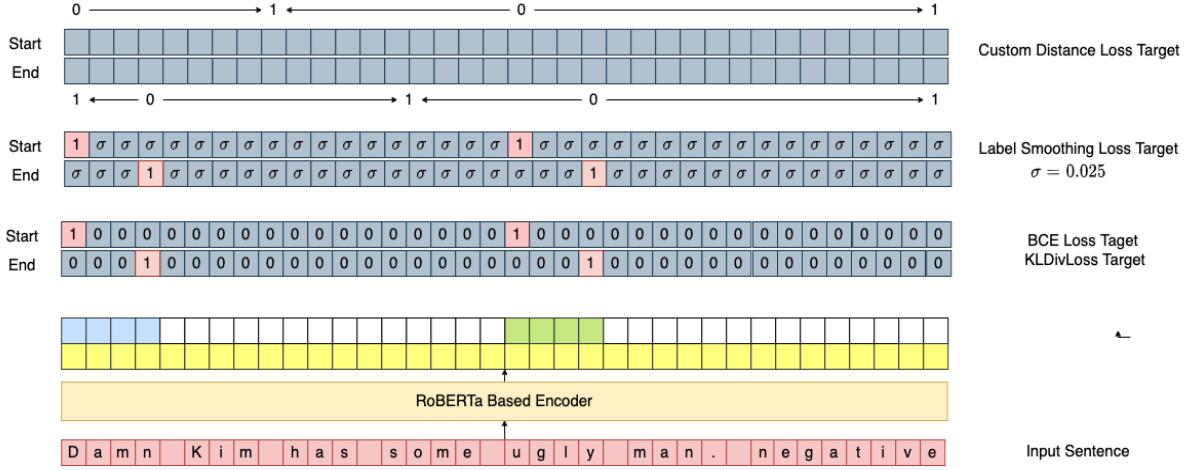


Figure 1: Start-to-end Tagging Framework

of trail, training and testing dataset can be shown in Table 1.

	Trail	Train	Test
Without span	43	485	394
With span	647	7454	1606
Total	690	7939	2000

Table 1: Datasets for SemEval-2021 Task 5

In this task, we apply the 5-fold cross-validation method and only use the official training data set for training and validating.

## 4.2 Evaluation Measure

To evaluate the responses of a system, we employ the F1 score, as in Martino et al. 2019. Let system  $A_i$  return a set  $S_{A_i}^t$  of character offsets, for parts of the post found to be toxic. Let  $G^t$  be the character offsets of the ground truth annotations of  $t$ . We compute the F1 score of system  $A_i$  with respect to the ground truth  $G$  for post  $t$  as follows, where  $||$  denotes set cardinality.

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|}$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|}$$

If  $S_G^t$  is empty for some post  $t$  (no gold spans are given for  $t$ ), we set  $F_1^t(A_i, G) = 1$  if  $S_{A_i}^t$  is also empty, and  $F_1^t(A_i, G) = 0$  otherwise. We finally

average  $F_1^t(A_i, G)$  over all the posts  $t$  of an evaluation dataset  $T$  to obtain a single score for  $A_i$ .

## 4.3 Experiments

The model is implemented using Pytorch (Paszke et al. 2019). We experiment with RoBERTa (Liu et al. 2019) based pre-trained language model as encoder, including RoBERTa-base-squad2 (Deepset), twitter-RoBERTa-base-sentiment (Barbieri et al. 2020) and DistillRoBERTa-base (Sanh et al. 2019). And we take the average of last two hidden layers’s weights as embedding. Our model is trained with AdamW (Loshchilov and Hutter 2017) optimizer with initial learning rate 0.00003 and weight decay coefficient 0.012. The max sequence length is 512 and dropout (Srivastava et al. 2014) rate is 0.5 to prevent our model from over fitting. And the threshold is set to 0.5. The final submission which scores 69.03 is equipped with both the binary cross-entropy loss, custom distance loss and voting ensemble mechanism.

## 4.4 Results and Analysis

In order to evaluate the effect of the custom loss function, we compare our approach with its variant.

Variant 1: The variant only use the cross-entropy loss.

Variant 2: The variant only use label smoothing loss.

Variant 3: The variant only use Kullback-Leibler divergence loss.

We take DistillRoBERTa-base as encoder for all of the above experiments. Table 2 show that the variant 2 model has the lowest score, which



Model	F1 score
Variation 1	0.6662
Variation 2	0.6347
Variation 3	0.6618
Our Model	<b>0.6754</b>

Table 2: Performance of Our System and Its Variants

may be caused by the enormous '0' label. Besides, the model with binary cross-entropy loss and custom distance loss obtains the best result. Thus we decide to use the model as our final ensembling element.

#### 4.5 Voting Ensemble

As mentioned above, we tried several RoBERTa based pre-trained language model as encoder. In this section, we will discuss the performance difference between them.

Encoder	F1 score
DistillRoBERTa-base	0.6754
RoBERTa-base-squad2	<b>0.6793</b>
twitter-RoBERTa-base-sentiment	0.6742

Table 3: Performance of Different Encoder

As shown in Table 3, we can find that the overall score's difference is slight. But when we take a closer look at the performance, the result on single example is different. And the RoBERTa-base-squad2 encoder achieved best result, which may be caused by the training method.

Text: "good side of trump? are you kidding me? trump has no good side all bad, he is divisive, a racist and bigot, pathological liar, scammer, tax cheat, sexual pervert,"

Golden Spans: ['pathological liar', 'scammer', 'sexual pervert']

DistillRoBERTa-base: ['racist and bigot', 'sexual pervert']

RoBERTa-base-squad2: ['racist', 'scammer', 'sexual pervert']

twitter-RoBERTa-base-sentiment: ['racist and bigot, pathological liar, scammer', 'sexual pervert']

As shown above, Complementing and correcting each other may improve the overall performance due to the difference. This is exactly what ensemble learning is good at. Ensembling of several models is widely used method to improve the performance of the overall system by combining predictions of

several models, such as as for they provide complementary information.

Considering this, we decide to apply the model ensemble methods, particularly the vote mechanism was applied. In which, if the number of occurrences of one index is bigger than 3 in the all above model's predictions, the index will be add to the final result, otherwise it will be exclude. The ensemble result obtains 69.03% F1 score on the test data set without any rule correction or dictionary based post process. Our model ranks in the top 10 among nearly 100 participating teams with slight lower (-1.8 and -1.73) than the top 1 (70.83%) and top 2(70.77%), respectively.

## 5 Conclusion and future work

In this paper, we propose a start-to-end tagging framework with custom distance loss function for SemEval-2021 Task 5. The performance of our model which is equipped with distance loss and voting mechanism better than its variants. But the distance loss target is assigned manually, which may have low generalization ability to different data set and task. We will try to improve its performance and apply this tagging scheme to other task in future work.

## References

- Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa Anke, and Leonardo Neves. 2020. [TweetEval: Unified benchmark and comparative evaluation for tweet classification](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1644–1650, Online. Association for Computational Linguistics.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Deepset. Farm. <https://github.com/deepset-ai/FARM>.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. [Bidirectional LSTM-CRF models for sequence tagging](#). *CoRR*, abs/1508.01991.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Ilya Loshchilov and Frank Hutter. 2017. [Fixing weight decay regularization in adam](#). *CoRR*, abs/1711.05101.

- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news articles](#). *CoRR*, abs/1910.02517.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#). *CoRR*, abs/1301.3781.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter](#). *CoRR*, abs/1910.01108.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. [Dropout: A simple way to prevent neural networks from overfitting](#). *Journal of Machine Learning Research*, 15(56):1929–1958.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. [Rethinking the inception architecture for computer vision](#). *CoRR*, abs/1512.00567.
- William Warner and Julia Hirschberg. 2012. [Detecting hate speech on the world wide web](#). In *Proceedings of the Second Workshop on Language in Social Media*, pages 19–26, Montréal, Canada. Association for Computational Linguistics.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2016. [Ex machina: Personal attacks seen at scale](#). *CoRR*, abs/1610.08914.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffenseEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. [Joint extraction of entities and relations based on a novel tagging scheme](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.

# HamiltonDinggg at SemEval-2021 Task 5: Investigating Toxic Span Detection using RoBERTa Pre-training

**Huiyang Ding**  
School of Information  
University of Michigan  
Ann Arbor, MI 48105  
huiyangd@umich.edu

**David Jurgens**  
School of Information  
University of Michigan  
Ann Arbor, MI 48105  
jurgens@umich.edu

## Abstract

This paper presents our system submission to task 5: Toxic Spans Detection of the SemEval-2021 competition. The competition aims at detecting the spans that make a toxic span toxic. In this paper, we demonstrate our system for detecting toxic spans, which includes expanding the toxic training set with Local Interpretable Model-Agnostic Explanations (LIME), fine-tuning RoBERTa model for detection, and error analysis. We found that feeding the model with an expanded training set using Reddit comments of polarized-toxicity and labeling with LIME on top of logistic regression classification could help RoBERTa more accurately learn to recognize toxic spans. We achieved a span-level F1 score of 0.6715 on the testing phase. Our quantitative and qualitative results show that the predictions from our system could be a good supplement to the gold training set’s annotations.

## 1 Introduction

Toxic messages remain a small but persistent part of online communications (Fortuna and Nunes, 2018; Jurgens et al., 2019). NLP methods have been developed to identify these comments, often relying on deep-language models (Vidgen et al., 2019). However, the part of the message that is specifically toxic is often unknown. Such information is useful not only for validating and explaining the judgments of models (Carton et al., 2018), but can also be useful for moderators to use when making decisions and working with these models in their deployment (Carton et al., 2020; Liu et al., 2021). This paper describes our model<sup>1</sup> and error analysis for SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021).

Our model uses a deep learning approach to identify which tokens are toxic. The approach

<sup>1</sup>The code is available at <https://github.com/davidjurgens/offensive-span-detection>.

is motivated by two strands of prior work showing (1) that large language models can effectively serve as sequence-to-sequence (seq2seq) models and (2) that pre-training on a similar task can improve downstream performance (Phang et al., 2018; Gururangan et al., 2020). Here, we treat the toxic-span detection tasks as a seq2seq task, where given a sequence of tokens, the model outputs per-token judgments of whether the token is in the toxic span. Given the limited training data for Task 5, we increase our training data by generating a silver-standard set of span judgments from LIME explanations (Ribeiro et al., 2016) from a model trained to recognize toxic and non-toxic language. These additional judgments are intended to help the model learn the basic span recognition task and identify general toxic language, before fine-tuning on the Task 5 data.

## 2 System Description

Our core system relies on a standard RoBERTa model (Liu et al., 2019) that is trained on a sequence-to-sequence task in two phases. The first phase pretrains the model with heuristically-created spans, gathered from Reddit comments labeled for their toxicity. The second phase fine-tunes this model on the organizer-provided data. Figure 1 shows the overview of the system. All the data used in the paper is in the English language.

### 2.1 Pretraining to Recognize Toxicity

To identify toxic spans, we hypothesize that pre-training the RoBERTa model on a similar task would lead to better downstream performance. Therefore we generate a similar dataset (silver dataset) to the training data (gold dataset) and heuristically label it with spans by using LIME (Ribeiro et al., 2016) on a toxicity classification task.

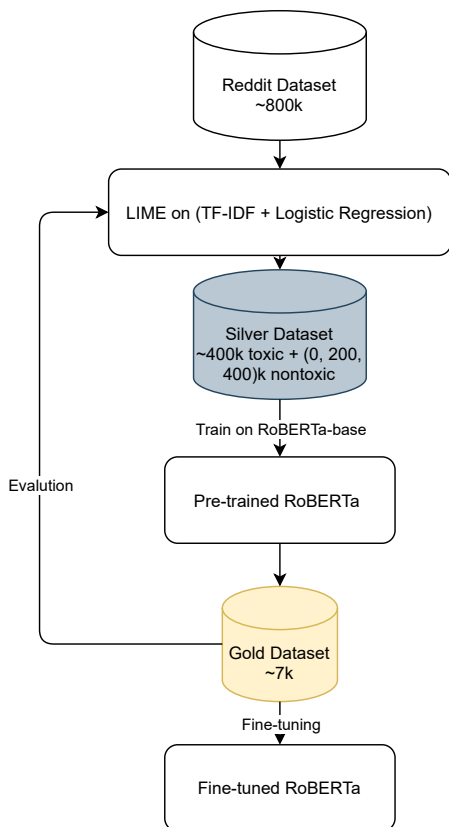


Figure 1: Diagram of our data and architecture. The central hypothesis tested is whether pre-training a RoBERTa model on machine-generated rationales for toxicity could improve performance.

**Data** Silver data was drawn from a sample of all Reddit comments made between January to June 2018. As social media data, these comments contain similar lexical and syntactic patterns as the social media comments data as the gold standard, which was made on the Civil Comments platform. Prior work has shown that pre-training RoBERTa models to recognize this type of social media data improve downstream performance (Nguyen et al., 2020). However, Reddit posts can vary substantially in their length. To avoid introducing confounding effects from pre-training a model on posts of substantially different lengths, we compute the Inter-quartile Range (IQR) of the lengths of Reddit comments and remove all comments identified as outliers. This process effectively removes very long or very short comments. Ultimately, the mean number of words in the training data and Reddit data are roughly similar:  $35.87 \pm 34.92$  words (mean and standard deviation) in the training data, compared with  $36.79 \pm 30.57$  in the Reddit data.

**Identifying Toxic Comments** The Reddit data contains a mix of toxic and non-toxic conversations, which we aim to use for training. To identify toxic conversations, we use the Perspective API to label all comments in the dataset (Wulczyn et al., 2017). The API returns a continuous score reflecting the degree of a comment’s toxicity. This toxicity score is then converted into a binary label to use in training a LIME model to generate rationales for why a comment is (or is not) toxic. We follow the insights from Hua et al. (2020) and set a threshold of 0.7, above which a comment is considered toxic and 0.3, below which the comment is non-toxic. These thresholds were intended to help create easy examples of toxic language for generating rationales as a way of scaffolding the learning for the downstream task. This process led to a labeled dataset of 288.5M comments with binary toxicity labels, of which 9.4% were labeled toxic.

### Generating Heuristically-Labeled Toxic Spans

Our final silver dataset is created by sampling comments from the larger labeled Reddit comments and using a LIME model to generate toxic span labels. LIME is a form of interpretable machine learning that explains the decisions of a classifier using a local approximation to identify which features led to a classification decision. Here, we use a simple logistic regression (LR) model trained on TF-IDF features and use LIME to generate a rationale of the classifier’s decision which identifies which words are contributing to the toxicity decision. The underlying LR model is trained on a balanced sample of 800K toxic/non-toxic comments (not the silver data). This balanced sample is derived in the same way as silver data. The model’s hyperparameters were tuned using 10-fold cross-validation, with the learning rate of 0.01 and strength of the regularization (C) at 1 under L1 loss. In a test of a held-out 200K instances, the model attained a binary F1 of 0.985.

Our silver data is created by generating LIME explanations using the trained LR classifier on a separate 800K comments balanced between toxic and non-toxic. This size is roughly 100x the Task-provided data. In generating explanations, LIME assigns local weights to each token on its weight to drive the correct prediction. To create toxic spans from these continuous-valued weights, we apply a threshold above which we consider the token as the toxic span. The threshold was identified by generating LIME explanations for all of 8629

documents from the Task’s training data and then choosing the threshold that maximized the Span F1 between the 8629 training documents’ toxic spans and the discretized LIME explanations, using a grid search with a step of 0.001 in [0.05,0.50]; the final threshold was set to 0.169.

## 2.2 Model Training

Our model uses a common RoBERTa base and differs according to which data the model is trained on. The pre-training setup trains a RoBERTa model on a seq2seq task where the input sequence of tokens generates a binary sequence denoting whether the input token was inside or outside of a toxic span. Due to the dataset sizes, pre-training was done for one epoch. The fine-tuning setup starts from either the off-the-shelf RoBERTa parameters or from a RoBERTa model initialized through silver-data pre-training. This model is trained in the same way as in pre-training on a binary seq2eq task using the Task-provided data. Models are fine-tuned for 10 epochs and parameters are chosen using the epoch with the best performance on the trial data.

In internal testing, we compared models that have been pre-trained, fine-tuned, or both, using varying amounts of silver data. All hyperparameter choices are reported in the Appendix A.

## 3 Results

Our best model attained a Span F1 of 0.672, and although close in score to the top result (0.708), was ranked 30 in the Task. Surprisingly, this best-performing model did *not* make use of the pre-training on silver data. To better understand the performance, we ran two follow-up analyses to test how different strategies for training affected performances and an error analysis for what were common themes in errors.

### 3.1 Does Pre-training Make a Difference?

In assessing the impact of pre-training, we analyze the submitted model along with five other models: (1) a fine-tuned model using a batch size of 8, (2) a pre-trained only model that makes no use of the Task data, and (3-5) pre-trained and fine-tuned models that use different amounts of silver data. The performances of all models are shown in Table 1.

For the initial comparison, we contrast the fine-tuned model (Table 1, Row 1; denoted FT) with the pre-trained and fine-tuned model on all silver

Model	Batch Size	Silver Data	F1
FT	8	N/A	<b>0.675</b>
FT	16	N/A	0.672 <sup>†</sup>
PT	8	400k/400k (1:1)	0.613
PT + FT	8	400k/0 (1:0)	0.660
PT + FT	8	400k/200k (2:1)	0.660
PT + FT	8	400k/400k (1:1)	0.659

Table 1: Performance at recognizing toxic spans (Span F1) for models trained on just the Task-provided training data (baseline), Pre-Trained (PT) on different amounts and ratios of silver data, and Fine-Tuned (FT) on training data. Ratios denote the number of non-toxic:toxic examples. <sup>†</sup> is the model submitted to the Task.

data (Table 1, Row 6; denoted PTFT). Both models agree on 1281 (65%) of the 2000 test instances. For these agreed cases, both models attain a Span F1 of 0.776—higher than either models regular performance. In these matching predictions, the ground-truth spans have an mean length of 1.13 tokens, mainly concentrated on commonly-labeled offensive words, like “morons”, suggesting that both models are adept at identifying overtly toxic words. In contrast, for the 21 test documents whose spans have  $\geq 5$  words, both models perform poorly with a Span F1 of 0.3489 for the FT model and 0.2077 for the PTFT model.

The FT model performs considerably better than PTFT model on documents with  $\geq 5$  tokens labeled as ground-truth spans, which is likely due to differences between the LIME-labeled data and the Task’s training data. For example, the LIME model generates spans  $\geq 5$  tokens in only 693 of the 800K silver context, suggesting LIME tends to give shorter toxic span labels.

This bias affects the downstream model performance in the test set where 29 of the 2000 test contexts have a span of  $\geq 2$  consecutive toxic words. In those contexts, the FT model achieves a mean Span F1 of 0.523 while the PTFT model has only 0.369. Indeed, the FT model produces spans (average span length: 19.0345) that are  $\sim 224\%$  longer than the PTFT model (average span length: 8.4828). This difference is more obvious than the overall prediction results, as seen in Table 2.

In the remaining cases where the FT and PTFT model predictions differ, the FT model has 296 predictions with a better Span F1 score, of which 216 predictions have longer span. For example, in test context 100, “Stupid is as stupid does Gump was right,” both of the “stupid” tokens are highlighted

Span Size	FT model	PTFT model
Characters	9.30 ± 6.44	7.05 ± 3.48
Tokens	2.14 ± 1.72	1.61 ± 1.00

Table 2: Differences in model prediction length shows that pretraining on LIME-generated toxicity rationales (PTFT) generally produces shorter spans at both span and token levels.

by the FT model, while the PTFT model only labels out the first “stupid”.

However, the tendency of the FT model to predict longer spans does not always yield higher performance. The PTFT model has 237 predictions with better Span F1 scores.

The FT model has a longer span prediction in the 202 of the 237 predictions, but the qualitative results are very different from the above-mentioned example. In multiple cases when the ground-truth spans are empty, the PTFT model can also predict empty spans. However, the FT model has a lower tendency to predict empty spans in those cases. For example, in non-toxic test context 3, “The parallels between the ANC and the Sicilian Mafia are glaring...”, the FT model labels “Sicilian” as toxic, while the PTFT model output is (correctly) empty.

Looking at contexts where there are no underlying toxic spans, the two models perform slightly differently. There are 394 out of 2000 test contexts with empty ground-truth spans. For those contexts, the FT model only gets a mean Span F1 score of 0.058, while the PTFT model gets 0.079.

In contrast to the FT model, the PTFT model has less-accurate predictions on the overtly/commonly toxic spans. For example, there are 430 total “stupid” or “stupidity” related words labeled as toxic by the ground-truth spans. The FT model is able to label 383/430 as toxic, while the PTFT model only labels 331/430. As we know, words like “stupid” can be more contextually-sensitive when compared to other common offensive words. They could be used in a toxicity-neutral way in many contexts. In the PTFT model’s pre-training phase, we fed 400,000 non-toxic documents for the RoBERTa model. These non-toxic documents supplied more non-offensive context for certain toxic words than the small-sized gold dataset. The extra contextual information learned by the pre-trained model can somehow decrease the performance of the PTFT model.

### 3.2 Common Themes in Errors

From the error analysis in the above section, we have noticed that the PTFT model does not perform well when it comes to predicting long toxic spans, empty toxic spans, and toxic phrases. With a deeper dive into the differences between PTFT model predictions and ground-truth spans, we can get a better sense about how is our PTFT model doing and what insights could be recommended for annotators.

In order to perform both qualitative and quantitative error analyses, we randomly sampled 200 test contexts where predictions from the PTFT model do not conform with ground-truth labels. The overall mean Span F1 score on those mispredictions is 0.289; compared to the random sample with a mean of 0.274. We categorize the mistake types from the sampled contexts and provide examples in Table 3.

**Model Errors** The first four categories (Category 1–4) in Table 3 demonstrate the mistakes made by the PTFT model and ground-truth annotations.

Category 1 shows where the PTFT model identifies valid toxic spans not present in ground truth, which accounts for 101 (50.5%) of the model errors in the 200 sampled contexts. In these cases, annotators marked nothing as toxic in 58 contexts. However, most of the overlooked toxic spans are overly-common toxic words like Examples 491 and 1374 in Table 3.

In comparison, in Category 2, there are 81 out of 200 sampled contexts with unmarked toxic span labels in the PTFT model output, in which the PTFT model produced an empty span or an incorrect span as toxic in four cases. Example 642 in Table 3 shows a typical case where a relative rare toxic word, “caca,” is overlooked by the model. In the remaining cases, the PTFT model has shorter predictions than the ground truth toxic spans, matching the low performance on predicting longer toxic spans (§3.1). In some cases, when multiple toxic spans exist in the same document, ground truth only labels one or two spans of them (e.g. Examples 1852 and 1486). While in other cases, ground truth would label more toxic spans (e.g. Example 346). The inconsistencies from ground truth annotations occur more when it comes to the multiple-word labeling, which will be described more in the next analysis for Category 5.

The missing labels from both scenarios are

Category	%	Examples
1. Toxic not labeled by ground-truth spans	50.5	491. this guy is an <b>idiot</b> . I'll disclose I'm male.... 1374. That is a bunch of horse <b>sh*t</b> . 1776. <b>Stupid</b> leftists. <b>Stupid</b> Democrats. I sincerely..
2. Toxic not labeled by the PTFT model	40.5	1852. That's because Floater <b>sucks</b> . That dudes voice is so <b>fucking</b> annoying! 1486. uhhh Hillary Clinton is a <b>serial killer</b> and <b>thief</b> 346. ...Drivers in Anchorage are lazy, <b>stupid</b> and <b>impatient</b> ... 642. What a load of <b>caca</b> !! ...
3. Not toxic but labeled by ground-truth spans	0.5	876. Ergo, America is overrun with the <b>insane</b> .
4. Not toxic but labeled by the PTFT model	13.5	273. Extremist <b>Muslims</b> want to kill infidels. Moderate Muslims cook the extremists a hot lunch. 1802. either you can afford to lose \$1300. in an hour or you're a <b>moron</b>
5. Inconsistent multiple-word/phrase annotation by ground-truth	4	773. Very true. Still <b>sick bastards</b> . 1496. Trump is an <b>impulsive idiot</b> . He will get us all killed. 1776. <b>Stupid</b> leftists. <b>Stupid</b> Democrats. I sincerely believe... 1447. Brooks, would you please join the <b>damn Democrat</b> party and be done with it ?
6. Inconsistent word annotation by ground-truth	41.5	968. ok then you dont use gasoline, plastic or such anything else right??? ya <b>hypocrite</b> bs <b>stupidity</b> as usual 348. ...Hawaii Democrats deny ordinary citizens their constitutional right to self-defense with firearms, including concealed carry... <b>Hypocrites</b> !
7. Inconsistent repeated word annotation by ground-truth	2	413. There is a difference between being tolerant and being <b>stupid</b> . She and her supporters want America to be <b>stupid</b> . 137. I cannot fathom the <b>stupidity</b> of McCain believing any cooperation from the ... The man must be the <b>stupidest</b> person in D.C.
8. Inconsistent sentence annotation by ground-truth	12.5	894. <b>You are a nut ball.</b> 1374.That is a bunch of <b>horse sh*t</b> .

Table 3: An error analysis of 200 predictions of our PTFT model relative to the ground-truth span. All of the sample contexts in the Examples column begin with the context index starting from 0. The first four categories show the differences in between the toxic spans labeling from PTFT model (shaded in blue) and the ground-truth span (bolded). The last four categories show the commonly-seen inconsistencies existing in the ground-truth annotations. Within the last four categories, examples are shown in pairs. Within each pair of examples, the upper example shows the suggested toxic spans based on the majority decisions made by the test context annotation (shaded in yellow) and the ground truth labeling decision by the annotators (bolded). In contrast, the lower example within each category shows where the consistency of annotation breaks compared to the upper example.

straight-forward in their formats and contexts. Besides, there are 20 contexts that both the ground-truth spans and PTFT model missed the toxic span partially or completely.

Generally, ground truth annotation seldom labels non-toxic spans as toxic (Category 3). On the contrary, it is common for our PTFT models to make mistakes on labeling non-toxic spans (Category

4). It happens mostly in the cases when the PTFT model misinterprets the context (e.g. Examples 273 and 1802).

**Inconsistencies in Ground Truth Labels** The last four categories (5–8) in Table 3 show common inconsistencies in annotation decisions, which we hope could aid in improving consistency in future work.

In Category 5, the standard for spans labeling is not consistent for which words are included in the toxic phrase. In some cases, when a sentence is fairly short (< 5 words) and contains toxic words, the ground truth annotation would label out the adjective used for describing the trailing noun (e.g. Example 1469). However, in other cases (e.g. Example 773), the standard would change by skipping the adjectives. Moreover, this inconsistent annotation also occurs in the case when the underlying nouns are almost the same (e.g. Examples 1447 and 1776).

Categories 6 and 7 comprise the majority of the inconsistencies in the annotation standards by the ground truth. These inconsistencies commonly manifest for frequent toxic words. For instance, in Examples 968 and 348, both of the "hypocrite(s)" should be toxic given the context and there is no more than one other toxic word within the document. The omission of common toxic words is the major source for this category. In many cases, the subtle variations of the document context would make it even harder to maintain a unified standard across different annotators. Hence, the introduction of some model-based labeling (or checking) could greatly improve the inconsistencies of this case.

In the sampled contexts, 25 documents consist of only one sentence. Annotators varied in how much of these contexts to label (Category 8), occasionally marking the entire sentence as offensive (20% of these single-sentence contexts), as in Example 894. However, in a few cases (4 of 25), annotators labeled nothing as toxic (e.g. Example 1374). Interestingly, 9 of 25 cases where ground truth either labels the entire sentence or nothing, our PTFT model is able to identify the toxic word(s), suggesting the model is still effective for short contexts.

## 4 Discussion and Future Work

Based on error analysis, our PTFT model suffers from low performance when generating predictions on non-toxic contexts or long toxic spans. A modified error function that rewards for edge-case sce-

narios can potentially improve the PTFT performance. Moreover, during the pre-training, we applied a simple-cutoff on the local weights to make labeling decisions for LIME explanations. The cutoff was determined solely based on evaluations with the Task data. If the LIME labeling could introduce more robust variants in the loss evaluation, the silver data span labeling might be more representative of the Task data's annotation logic.

Through comparing silver data with the gold data, we find the toxicity of some words is influenced by the broader linguistic environment. While the silver and gold data both consist of online comments, their time spans and topics are very different. The gold data uses contexts from 2015-2017 and has a concentration on political news; while the silver data covers 6 months of 2018 with no focused topics. Our qualitative analysis finds that the addition of non-toxic examples in the silver data influenced the model to consider overly common toxic words less toxic than they were in the gold data. Future work is needed to identify the optimal ratio of the toxic and non-toxic samples and to address domain/register differences in the data.

Last, the current approach could invite several natural improvements. For example, in the pre-training phase, we used TF-IDF embedding and logistic regression for the base of LIME explanations. This combination was chosen for its efficiency in the LIME training phase. However, many other embedding and model combinations rendered much better classification results, which may generate better rationales for pre-training.

## 5 Conclusion

We presented our system for SemEval-2021 Task 5 on Toxic Span Prediction. Our initial approach used explainable machine learning (LIME) to generate a heuristically labeled span dataset, which was used to pre-train a RoBERTa model to recognize toxic spans. However, our results show that when fine-tuned on the task data, the resulting model generates slightly shorter explanations and ultimately performs slightly worse than a model trained only on the Task's training data—likely due to bias towards shorter spans generated by LIME. In our subsequent error analysis, we show that the majority of our model's errors (50.5%) are associated with missed annotations in the ground truth, suggesting that actual model performance may be higher in practice.



## Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No 185022.

## References

- Samuel Carton, Qiaozhu Mei, and Paul Resnick. 2018. Extractive adversarial networks: High-recall explanations for identifying personal attacks in social media posts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3497–3507.
- Samuel Carton, Qiaozhu Mei, and Paul Resnick. 2020. Feature-based explanations don’t help people detect misclassifications of online toxicity. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 95–106.
- Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):85.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.
- Yiqing Hua, Thomas Ristenpart, and Mor Naaman. 2020. Towards measuring adversarial twitter interactions against candidates in the US midterm elections. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 14, pages 272–282.
- David Jurgens, Eshwar Chandrasekharan, and Libby Hemphill. 2019. A just and comprehensive strategy for using NLP to address online abuse. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Han Liu, Vivian Lai, and Chenhao Tan. 2021. Understanding the effect of out-of-distribution examples and interactive explanations on human-ai decision making. *arXiv preprint arXiv:2101.05303*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. *BERTweet: A pre-trained language model for English tweets*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.

Hyperparameter	Value
learning rate	5e-05
number of epochs	1
weight decay	0
model	RoBERTa-base
dropping probability of the dropout layer	0.5
learning rate scheduling	linear
optimizer	AdamW
warmup steps	300
random seed	42

Table 4: Hyperparameters of pre-trained model

Hyperparameter	Value
learning rate	5e-5
number of epochs	10
weight decay	1e-2
model	RoBERTa-base
dropping probability of the dropout layer	0.5
learning rate scheduling	linear
optimizer	AdamW
warmup steps	0
random seed	42

Table 5: Hyperparameters of fine-tuned model

- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. SemEval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.
- Bertie Vidgen, Alex Harris, Dong Nguyen, Rebekah Tromble, Scott Hale, and Helen Margetts. 2019. Challenges and frontiers in abusive content detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the Web Conference*.

## A Hyperparameter Settings

Tables 4 and 5 respectively show the hyperparameter settings for the fine-tuned (FT) model and the model pre-trained on silver data and then fine-tuned on the training data (PTFT).

# WVOQ at SemEval-2021 Task 6: BART for Span Detection and Classification

Cees Roele  
cees.roele@gmail.com

## Abstract

Simultaneous span detection and classification is a task not currently addressed in standard NLP frameworks. The present paper describes why and how an EncoderDecoder model was used to combine span detection and classification to address subtask 2 of SemEval-2021 Task 6.

## 1 Introduction

Task 6 of SemEval-2021 studies the detection of persuasion techniques (Dimitrov et al., 2021). The task considers English language memes, which in subtasks are to be classified, divided into classified fragments having a begin and end, and classified when text is combined with images.

Of the three subtasks described in the paper, the present paper primarily addresses resolving subtask 2:

Given only the "textual content" of a meme, identify which of the 20 techniques are used in it together with the span(s) of text covered by each technique. This is a multilabel sequence tagging task.

The figure below illustrates span detection and classification for three technique classes for a meme.

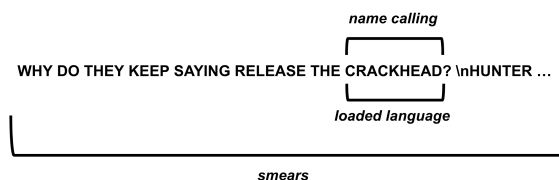


Figure 1: Span detection and classification: overlapping spans and spans extending over multiple sentences

In the above figure the ellipsis at the end of the sentence denotes the continuation of the second

sentence. Note that *loaded language* and *name calling* both apply to the same span, that is, the word "CRACKHEAD". Note furthermore, that the span for *smears* overlaps with both of these spans and ranges over more than one sentence.

The present paper describes a novel approach to resolving these requirements by generating XML-like start and end tokens to delineate spans. The following illustrates this for the message in figure 1.

```
<SMEARS>
WHY DO THEY KEEP SAYING RELEASE THE
<LOADED-LANGUAGE>
<NAME-CALLING>
CRACKHEAD
</NAME-CALLING>
</LOADED-LANGUAGE>
? "n HUNTER ...
</SMEARS>
```

It attained an F1 score on the test set that is about in the middle of the baseline and the highest ranking score.

The choice of this approach of generating markup to identify spans was made on the basis that it was technically possible, easily understandable at a behavioral level of input and output, and using a model that is pre-trained for dealing with spans. The aim was not so much to attain the highest score as to explore how effective this approach is in a proof-of-concept and what problems need to be overcome to bring it to good performance.<sup>1</sup>

## 2 Background

Propaganda messages are constructed using specific rhetorical techniques. The current task is to identify within a message in what fragment a particular technique is invoked.

<sup>1</sup>The code for the described system is available at: <https://github.com/ceesroele/SemEval-2021-Task-6>.

A simpler task is to identify fragments of a message in which any propaganda technique is used. Effectively, this comes down to classifying any part of a message as being either propaganda or not. It is a sequence labeling problem that can be resolved for example using a BIO tagging format, where BIO stands for Begin, Inside, and Outside. To classify a span of tokens as propaganda we can use B-PROP to designate the begin of the span, I-PROP to indicate the token being inside the earlier begun span, and O to designate a token not being part of a span. (Chernyavskiy et al., 2020).

For our case this approach can be extended by adding new labels for each technique, e.g. B-SMEARS, I-SMEARS, B-NAME-CALLING, B-LOADED-LANGUAGE, and so on for all twenty technique classes. But looking at figure 1 we see that if CRACKHEAD is a token, we have to simultaneously label it as I-SMEARS, B-NAME-CALLING, and B-LOADED-LANGUAGE. The extension of the approach by just adding labels is not applicable to our situation in which spans can overlap.

One solution for this problem is to retain the assumption that each input token is to be tagged, but add virtual depth. This approach was taken for the PPropaganda persuasion Techniques Analyzer (PRta) (Da San Martino et al., 2020). It is based on an architecture where each input token maps to as many output tokens as there are technique classes, plus one extra for *no technique*. Additionally, it uses a complementary output indicating confidence of any propaganda technique being present at the sentence level, which is used as a gate for predicting the presence of any specific techniques.

The sequence labeling method described at the beginning of this section is effectively a sequence-to-sequence translation, where the input and output sequence consist of the same number of tokens. This allows us to match input with output based on position. To generate a marked up version of a message we need to allow an output sequence to have a length that differs from the input sequence.

By using an EncoderDecoder model we can generate arbitrary transformations of an input message including changing its length. This can be used for abstractive dialogue, question answering, and summarization. A state of the art EncoderDecoder model is BART, a denoising autoencoder built with a sequence-to-sequence model. (Lewis et al., 2020).

BART uses a standard Transformer-based neural machine translation architecture to couple a bidirectional encoder with a left-to-right decoder. Pre-training BART was done by first corrupting text with an arbitrary noising function and then training a sequence-to-sequence model to reconstruct the original text.

## 2.1 Data

There are two datasets available for task 6. The first is the Propaganda Techniques Corpus (PTC) dataset from SemEval-2020 Task 11. It consists of about 550 English language news articles in which spans - defined by begin and end positions - have been annotated with one out of 18 propaganda techniques. In practice a number of these techniques have been combined. For example, the three techniques *whataboutism*, *straw men*, and *red herring* have been conflated into the single label *whataboutism, straw men, red herring*. As a result, the dataset has effectively been annotated with 14 labels. Moreover, these composite labels don't identify individual labels in the 2021 dataset, which makes them unsuitable for training. That leaves only 12 usable labels in the PTC.

The 2021 dataset consists of about 660 English language memes. These are short texts consisting of mostly short sentences and relatively many uppercase characters. Here fragments have been identified by start and end indexes and are labeled with one of a total of 20 classes. The differently labeled fragments may overlap, that is, a certain span of text may belong to fragments belonging to different classes.

The table below shows the number of fragments per dataset, the average number of words per fragment, the number of fragments spreading over more than one sentence, and the relative number of uppercase characters in fragments ( $\text{upper} / (\text{upper} + \text{lower})$ ).

Dataset	Spans	Words	> 1	Upper
PTC 2020	5610	8.6	223	0.04
Memes 2021	1497	7.6	224	0.53
Total set	7107	8.4	447	0.14

Table 1: Data

Regarding the data, we make the following observations:

- For 8 of the classes there is data only in the relatively small 2021 memes dataset, which with

many short sentences and a lot of uppercase is structurally different from the PCT 2020 dataset

- For some classes as much as half the characters in their fragments are in uppercase
- The median number of words in a fragment significantly varies per class. E.g. *smears*, *causal oversimplification*, and *whataboutism* have median numbers of words of respectively 16, 20, and 25, while *name calling/labeling*, *loaded language*, and *repetition* have median numbers of words of respectively 3, 2, and 1.
- The median number of sentences by fragments is 1 and for a handful of classes 2.

The above findings will inspire a number of choices specified in the Experimental Setup below.

## 2.2 Pre-training is key

The success of language models like BERT (Devlin et al., 2019) derives in great part from a division of labor and domain. In the first step, a model is trained on a large body of unmarked data. This results in a model that has many linguistic relations represented in its weights, but that by itself is of little use. In the second step, that resulting pre-trained model is fine-tuned with data from a specific domain.

Given the comparative smallness of the two datasets at our disposal, leveraging pre-training can be expected to greatly enhance the quality of predictions.

However, it is worth considering what the pre-training entails. Take BERT. It was trained in part on English Wikipedia articles. But now we are looking at memes full of uppercase characters, containing persuasion techniques that we hope are not used in Wikipedia. Said differently, the data the model was pre-trained on might not be representative for our domain.

More abstract, but no less important, is the method of pre-training. Is the used method of Masked Language Modeling (MLM) supporting our task? We are interested in spans of text, possibly running across multiple sentences. Besides next sentence prediction, BERT's methodology primarily consists of replacing a percentage of individual tokens with a mask token. However effective this may be, it is not optimized for spans.

SpanBERT (Joshi et al., 2020) is effectively BERT trained with a different masking method:

- mask contiguous random spans, rather than random tokens, and
- train the span boundary representations to predict the entire content of the masked span, without relying on the individual token representations within it

SpanBERT outperforms BERT substantially on span selection tasks such as question answering and coreference resolution.

The present paper concerns a specific implementation for span detection and classification. Understanding pre-training helps us understand both why the presented system has a certain success and what its limitations are.

## 3 System overview

### 3.1 Generating markup

As sketched in the Background section, the problem we need to resolve is how to simultaneously represent a span of text and one of a multitude of labels. Our solution is to step away from attempts to map onto a classification structure and instead regenerate the original text, but now with XML-like markup to indicate the start, end, and class of each fragment.

We regenerate the input text using an EncoderDecoder model. Popularly expressed, it reads a text, and then generates a sequence of words. In order to add our markup for fragments we need two help functions, let's call them encipher and decipher. The encipher function takes text plus metadata on fragments and converts this into a string with XML-like markup. We need this to create our training data. The decipher function takes a string including XML-like markup and extracts metadata in the form of start, end, and class from it.

For each of the labels, the names for our technique classes, we create a start tag and an end tag. In order to let the tokenizer treat them each as single tokens, we add all these tags as tokens to the tokenizer.

### 3.2 Using the BART EncoderDecoder model

In principle it is possible to implement Encoder and Decoder on the basis of taking a pre-trained model for each, e.g. RoBERTa for the Encoder and BERT for the Decoder. Finding from a single trial was that in such a setup training went very slow and outcome was dissatisfactory. Instead we

selected BART (Lewis et al., 2020) as an integrated EncoderDecoder model.

BART has a number of pre-training methods that are of interest in trying to understand its performance. Only the first one is part of the pre-training methodology of BERT.

- **token masking**, like BERT
- **token deletion**, random tokens are deleted and the model must decide which positions are missing tokens
- **text infilling**, a number of spans with varying lengths are sampled and replaced with a single mask token. Note that this is different from SpanBERT pre-training where each token of the span is replaced with a mask token.
- **sentence permutation**, sentences are shuffled in random order
- **document rotation**, a token is randomly chosen and the document is rotated to start with that token

We will get back to this when we evaluate the result.

### 3.3 Easy to generalize

Recuperating, we initialize the model by adding start and end tags for each technique class to the tokenizer. We use encipher to create marked up versions of input texts to train the model. To obtain fragments for given inputs we must decipher generated marked up texts to extract meta-data. Besides having markup, the generated text may be different from the input. Directly deriving span positions from the markup tags leads to errors when that happens. This is to some degree mitigated by using an algorithm that searches for the best place of the tag in the original input string.

The novelty of the described system is in using a standard EncoderDecoder model to generate markup. No special architectural changes were made, no domain dependencies were introduced, and only minor pre- and postprocessing is done. It is therefor easy to turn the system into a general purpose span detection and classification system.

## 4 Experimental setup

### 4.1 Data and Training

The articles of the PTC 2020 dataset were reduced to smaller segments on the basis of fragments.

```
for each fragment:
    take all covering sentences
    while another fragment overlaps ..
        .. with any sentence in the segment
    add fragment and those sentences
```

Any sentences remaining, that is, not covered by any fragment, were ignored. The memes of the 2021 dataset were not split.

Mixing the 2020 and 2021 datasets for a single training run led to worse results than having a staged training of first the PTC 2020 data as pre-training and then the 2021 memes data as fine-tuning. For training the datasets were split train:dev:test as 70:20:10. Training was done with a batch size of 8 for 25 epochs.

### 4.2 Framework

The system uses the Seq2SeqModel of Simple Transformers<sup>2</sup>, a task-oriented framework built on top of Hugging Face Transformers<sup>3</sup>. It uses the Hugging Face pre-trained BART model identified with model type "bart" and model name "facebook/bart-base". This is a model consisting of 6 encoder and decoder layers, 16-heads, and 139M parameters.

### 4.3 Configuration

Where training was done mostly with default settings, text generation required improved settings. We want enough tokens in the output for the full input plus markup, we want a relatively low penalty on length, to compensate for the previous setting, we want a relatively high penalty on repetition, and we perform a beam search. Experimentally, we came to the following settings as being optimal:

Parameter	Value
max_length	200
length_penalty	0.4
repetition_penalty	2.0
do_sample	True
num_beams	3
top_p	0.8

Table 2: Seq2SeqModel configuration

<sup>2</sup>See: <https://simpletransformers.ai/>. The used version is 0.60.6. To be able to add begin and end markers as tokens to the seq2seq model a modification was made. It can be found in the github repository for the system discussed here, referred to in the first footnote.

<sup>3</sup>See: <https://huggingface.co/transformers/>. The used version is 4.3.2

## 5 Results

The system’s F1 score of 0.268 on subtask 2 on the test set scores about in the middle between the baseline and the highest ranking score.

Rank	Team	F1 score	Precision	Recall
1	Volta	.482	.501	.464
5	WVOQ	.268	.243	.299
	baseline	.010	.034	.006

Table 3: Subtask 2 scores on the test set

Looking at errors we made the following observations:

- Beginning and end tags in the generated text regularly don’t match.
- Generated text contains changed words and even added words, which leads to faulty identifications of spans.

Why does this happen? First, beginning and end tags are introduced as new tokens in the relatively small datasets we fine-tune with. Transformer models have no notion of syntactic connection between them and standard BART has not been pre-trained to relate these tokens correctly. Second, through its pre-training methodology BART is geared towards relative “freedom” in filling in spans. That’s what makes it suitable for summarization and question-answering. But what we need for markup generation is almost verbatim regeneration of the input.

## 6 Conclusion

The described system for span detection and simultaneous classification offers a proof-of-concept for a novel approach to sequence tagging based on generating a version of a message with markup for labels. Its F1 score on the leaderboard is in the middle between the baseline and the top score.

Drawback of the approach is that two types of systemic errors are introduced: tags lacking a matching tag, and tokens generated that are not in the original message. These are not resolved by fine-tuning the model and they cannot be addressed with the standard configuration parameters of message generation in the sequence-to-sequence model.

Future research should aim at resolving these systemic errors. Matching tags could be addressed through changes in the decoder’s generation algorithm. Having the tokens in the output be the same

as those in the input could be improved by amending the loss function for fine-tuning training of the model.

Only when these two issues are resolved will further optimization of the approach be worth investing effort in.

## References

- Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. [Aschern at SemEval-2020 task 11: It takes three to tango: RoBERTa, CRF, and transfer learning](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1462–1468, Barcelona (online). International Committee for Computational Linguistics.
- Giovanni Da San Martino, Shaden Shaar, Yifan Zhang, Seunghak Yu, Alberto Barrón-Cedeño, and Preslav Nakov. 2020. [Prta: A system to support the analysis of propaganda techniques in the news](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 287–293, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Feroz Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at SemEval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval ’21*, Bangkok, Thailand.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. [SpanBERT: Improving pre-training by representing and predicting spans](#). *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. [BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

# HumorHunter at SemEval-2021 Task 7: Humor and Offense Recognition with Disentangled Attention

Yubo Xie, Junze Li, and Pearl Pu

School of Computer and Communication Sciences  
École Polytechnique Fédérale de Lausanne, Switzerland  
{yubo.xie, junze.li, pearl.pu}@epfl.ch

## Abstract

In this paper, we describe our system submitted to SemEval 2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense. The task aims at predicting whether the given text is humorous, the average humor rating given by the annotators, and whether the humor rating is controversial. In addition, the task also involves predicting how offensive the text is. Our approach adopts the DeBERTa architecture with disentangled attention mechanism, where the attention scores between words are calculated based on their content vectors and relative position vectors. We also took advantage of the pre-trained language models and fine-tuned the DeBERTa model on all the four subtasks. We experimented with several BERT-like structures and found that the large DeBERTa model generally performs better. During the evaluation phase, our system achieved an F-score of 0.9480 on subtask 1a, an RMSE of 0.5510 on subtask 1b, an F-score of 0.4764 on subtask 1c, and an RMSE of 0.4230 on subtask 2a (rank 3 on the leaderboard).

## 1 Introduction

Humor, appreciated by people with almost any age or cultural background, is perhaps one of the most fascinating human behaviors. Besides providing entertainment, humor can also be beneficial to mental health by serving as a moderator of life stress (Lefcourt and Martin, 2012), and plays an important role in regulating human-human interaction. As Reeves and Nass (1996) have pointed out, people respond to computers in the same way as they do to real people, which indicates that modeling humor computationally could bring positive effects in human-computer interaction (Nijholt et al., 2003). Despite being universal to human beings, the extent to which people find something humorous varies according to one’s age, gender, or socio-economic

status, making humor a highly subjective experience. This poses many challenges to the field of computational humor. Abundant research has been done to enable computers to automatically decide whether humor is entailed in a given piece of text. Early work (Mihalcea and Strapparava, 2005; Mihalcea et al., 2010) uses manually engineered features to recognize humor in text, while more recent work (Chen and Soo, 2018; Weller and Seppi, 2019) adopts deep learning approaches and pre-trained language models.

SemEval 2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense (Meaney et al., 2021) aims at detecting and rating humor as well as offense in short English text. There are four subtasks involved. Subtask 1a is a binary classification task, predicting if the text would be considered humorous for an average user. Subtask 1b is a regression task and predicts the humor rating of the text if it is considered humorous. Subtask 1c is again a binary classification task and predicts whether the humor rating is controversial, whose ground-truth label is decided based on the variance of the annotators’ ratings. This task also involves offense detection. Subtask 2a predicts how offensive the text is for a general user. All the regression subtasks have scores ranging from 0 to 5.

In this paper, we present our system submitted to SemEval 2021 Task 7. We followed the architecture of DeBERTa (He et al., 2020), an improved version of BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) by using two novel techniques: disentangled attention and decoding enhanced masking. We mainly relied on the disentangled attention mechanism, where the attention weights of the input words are calculated based on their content vectors and relative position vectors. For the four subtasks, we used the same base structure and the only difference is at the output layer, where the classification tasks have two out-

put units and the regression tasks only have one. The pre-trained DeBERTa model has two variants that differ in size. During the evaluation phase, the large version achieved an F-score of 0.9480 on subtask 1a, an RMSE of 0.5510 on subtask 1b, an F-score of 0.4764 on subtask 1c, and an RMSE of 0.4230 on subtask 2a (rank 3 on the leaderboard). In addition, we also experimented with the BERT and RoBERTa models as our baselines, and found them generally under-performed by DeBERTa. Our code has been made publicly available.<sup>1</sup>

## 2 Related Work

Mihalcea and Strapparava (2005) used several human-centric features such as alliteration and synonym to recognize humor in one-liners. Mihalcea et al. (2010) approached the problem by calculating the semantic relatedness between the set-up and the punchline. Morales and Zhai (2017) proposed a generative language model and leveraged background text sources to identify humor in Yelp reviews. Liu et al. (2018) proposed to model sentiment association between elementary discourse units and designed features based on discourse relations. Xie et al. (2020) calculated the uncertainty and surprisal of the set-up and the punchline according to the incongruity humor theory, which were found useful in humor recognition. Recent work also developed neural network based models to recognize humor in text. Chen and Lee (2017) and Chen and Soo (2018) adopted convolutional neural networks, while Weller and Seppi (2019) used a Transformer architecture.

## 3 Dataset

SemEval 2021 Task 7 provides three datasets: the training set (8,000), the validation set (1,000), and the final test set (1,000). Table 1 summarizes the statistics of the three datasets, and lists the respective information of humorous (positive) and non-humorous (negative) examples. Each example is a piece of English text accompanied by four features: `is_humor` (subtask 1a), `humor_rating` (subtask 1b), `humor_controversy` (subtask 1c), and `offense_rating` (subtask 2a). For subtask 1b and 2a, the labels range from 0 to 5. Table 2 gives two samples, one being humorous and the other non-humorous.

<sup>1</sup><https://github.com/yuboxie/semEval-2021-task-7>

	Train	Validation	Test
# positive	4,932	632	615
Avg # tokens	24.48	22.04	26.14
# negative	3,068	368	385
Avg # tokens	25.95	26.12	29.36
# total	8,000	1,000	1,000
Avg # tokens	25.05	23.54	27.38

Table 1: Statistics of the provided datasets. Here the respective information of humorous (positive) and non-humorous (negative) examples are also listed.

For subtask 2a, whose goal is to predict the offense rating of the input text, we also visualize top 200 frequent unigrams for examples with offense rating  $\geq 2$  and  $< 2$ , respectively, illustrated as two word clouds (Figure 1a and Figure 1b). As we can observe, Figure 1a contains words that are expected to appear in offensive text, usually targeting at a specific group of people (e.g., “black”, “gay”, “chinese”, “muslim”, etc.), while Figure 1b contains more ordinary words, which generally do not imply offense.

## 4 System Overview

With the increasingly powerful neural networks such as the Transformer (Vaswani et al., 2017), the performance on many downstream NLP tasks has been greatly improved by fine-tuning large pre-trained language models on smaller but task-specific datasets. Traditional Transformer-based language models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) use absolute positional embeddings in the input layer, which are added up with the word embeddings and serve as the input to the following Transformer layers. The self attention weights between the tokens are calculated solely based on their hidden representations. However, recent work (Shaw et al., 2018; Dai et al., 2019) has shown that relative position representations are more effective for NLP tasks.

Our system leverages the disentangled attention mechanism from the DeBERTa model (He et al., 2020), where the attention weights between input tokens are calculated based on their content vectors as well as their relative positions. As shown in Figure 2, for each Transformer layer,  $H_i$ ’s are the input representations from last layer, and  $H_i^o$ ’s are the output representations after applying the self attention. Instead of using absolute positional



text	is_humor	humor_rating	humor_controversy	offense_rating
Here’s a FedEx joke - actually, you’ll get it tomorrow.	1	3.21	0	0
When humans make mistakes, it doesn’t mean they’re evil, it means they’re human.	0	-	-	0.1

Table 2: Two samples from the training set.



(a) Word cloud of examples with offense rating  $\geq 2$



(b) Word cloud of examples with offense rating  $< 2$

Figure 1: Word clouds of the data according to the offense rating.

embeddings at the input layer, we create a relative positional embedding table, which is shared across all layers, to represent the relative position between token  $i$  and token  $j$ . More specifically, the index of the relative position between token  $i$  and  $j$  is defined as

$$\delta(i, j) = \begin{cases} 0 & \text{if } i - j \leq -k, \\ 2k - 1 & \text{if } i - j \geq k, \\ i - j + k & \text{otherwise,} \end{cases} \quad (1)$$

where  $k$  is the maximum distance we consider. Similar to normal Transformer attention mechanism, the content representations  $H$  and the relative position representations  $P \in \mathbb{R}^{2k \times d}$  are transformed to queries, keys, and values:

$$\begin{aligned} Q^c &= HW_q^c, K^c = HW_k^c, V^c = HW_v^c, \\ Q^p &= PW_q^p, K^p = PW_k^p. \end{aligned} \quad (2)$$

Then, the attention weight  $A_{ij}$  between token  $i$  and token  $j$  are calculated as follow:

$$A_{ij} = Q_i^c K_j^{cT} + Q_i^c K_{\delta(i,j)}^{pT} + K_j^c Q_{\delta(j,i)}^{pT}. \quad (3)$$

When aggregating the input representations  $H$ , we apply a scaling factor  $1/\sqrt{3d}$  to obtain the output representations  $H^o$ :

$$H^o = \text{softmax}\left(\frac{A}{\sqrt{3d}}\right)V^c. \quad (4)$$

For subtask 1a and 1c, which are binary classification tasks, we use softmax output layer and

cross entropy loss. For subtask 1b and 2a, which are regression tasks, we use mean square error as the loss function. Otherwise, the base structure is the same, and we initialize the model with the pre-trained DeBERTa weights.

## 5 Experimental Setup

We evaluated and compared our system with several baselines on the provided dataset, whose statistics are provided in Section 3. In this section, we are going to elaborate the setup of our experiment.

### 5.1 Baselines

In our experiment, we consider the following approaches as our baselines:

- **Bag of words (BoW)**. In this approach, we neglect the order of the input tokens, and simply add up the word embeddings of the tokens to form the vector representation of the input text. We implemented logistic regression for subtask 1a and 1c, and linear regression for subtask 1b and 2a, using the 300d GloVe word embeddings (Pennington et al., 2014).
- **Convolutional neural network (CNN)**. Convolutional neural networks have been widely adopted in computer vision and image recognition. When applied to NLP tasks, the input is a 2D matrix with each row being the word embeddings of the respective token, and the convolution is operated along the rows, with a

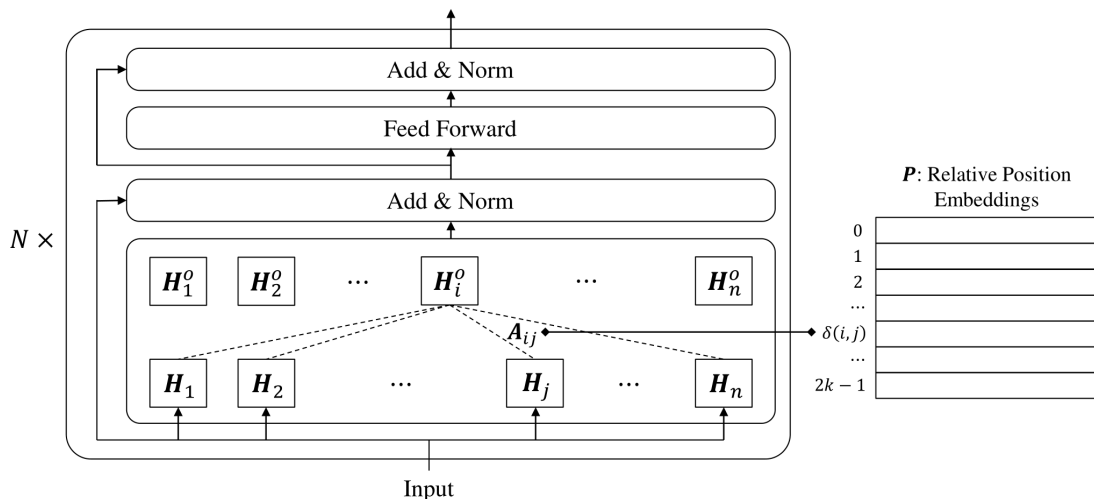


Figure 2: An illustration of the model architecture.

	Subtask 1a			Subtask 1b	
	Precision	Recall	F-Score	Accuracy	RMSE
BoW	0.7884/0.8141	0.7712/0.8067	0.7778/0.8099	0.7990/0.8220	0.5433/0.5617
CNN	0.8289/0.8524	0.8221/0.8485	0.8252/0.8503	0.8390/0.8590	0.6661/0.6399
Bi-LSTM	0.8340/0.8620	0.8438/0.8610	0.8381/0.8615	0.8470/0.8690	0.5645/0.5504
BERT (base)	0.9061/0.9119	<b>0.9462/0.9593</b>	0.9257/0.9350	0.9040/0.9180	0.4994/0.5402
BERT (large)	0.9246/0.9442	0.9320/0.9350	0.9283/0.9395	0.9090/0.9260	0.5099/0.5500
RoBERTa (base)	0.9398/0.9469	0.9383/0.9577	0.9390/0.9523	0.9230/0.9410	0.5259/0.6320
RoBERTa (large)	0.9597/0.9515	0.9415/0.9561	0.9505/ <b>0.9538</b>	0.9380/ <b>0.9430</b>	0.4994/ <b>0.5326</b>
Our system (base)	0.9463/0.9521	0.9209/0.9382	0.9334/0.9451	0.9170/0.9330	0.4978/0.5456
Our system (large)	<b>0.9707/0.9604</b>	0.9446/0.9463	<b>0.9575/0.9533</b>	<b>0.9470/0.9430</b>	<b>0.4923/0.5538</b>

Table 3: Performance of subtask 1a and 1b on the validation / test set.

fixed window size. We follow the CNN model in the work of [Chen and Lee \(2017\)](#), which includes an extra highway layer before the final fully connected layer, allowing shortcut connections with gate functions.

- **Bidirectional long short-term memory (Bi-LSTM).** LSTM ([Hochreiter and Schmidhuber, 1997](#)) has shown to perform quite well in handling sequential inputs, making it suitable for many NLP tasks. Bidirectional LSTM incorporates two LSTMs, one in the forward direction and the other in the backward direction, thus better modeling the context. In this approach, we use a Bi-LSTM with hidden size 200 and one hidden layer.
- **BERT.** BERT ([Devlin et al., 2019](#)) is a deep bidirectional Transformer pre-trained on BooksCorpus and English Wikipedia, with two training objectives: (1) masked language model, where some of the input tokens are randomly masked and are to be recovered by the model; (2) next sentence prediction, where

the goal is to predict if the input second sentence follows the first one. By fine-tuning the pre-trained BERT, the performance of a wide range of NLP tasks can be largely improved, compared with previous models such as LSTMs.

- **RoBERTa.** RoBERTa ([Liu et al., 2019](#)) is an optimized version of BERT, which was trained on bigger datasets and longer sequences. In addition, the next sentence prediction objective was removed, which was found to slightly improve the performance of downstream tasks. RoBERTa reportedly achieved better results than BERT on benchmarks such as GLUE, RACE and SQuAD.

## 5.2 Implementation

All the Transformer-based models in the experiment have two variants that differ in model size. The base version has 12 Transformer layers, 768 hidden units, and 12 multiheads. The large version has 24 Transformer layers, 1024 hidden units,

	Subtask 1c				Subtask 2a
	Precision	Recall	F-Score	Accuracy	RMSE
BoW	<b>0.5539/0.5585</b>	0.5539/0.5584	0.5538/0.5584	0.5538/ <b>0.5626</b>	0.9418/0.7207
CNN	0.5052/0.5084	0.5051/0.5084	0.5012/0.5055	0.5032/0.5057	0.8238/0.6913
Bi-LSTM	0.4907/0.4908	0.4907/0.4919	0.4905/0.4817	0.4905/0.5089	0.7825/0.6666
BERT (base)	0.5455/0.4924	0.5649/0.4659	0.5550/0.4788	<b>0.5585</b> /0.5398	0.5681/0.5228
BERT (large)	0.5013/0.4891	0.6071/0.5627	0.5492/0.5233	0.5142/0.5350	0.5550/0.5022
RoBERTa (base)	0.4873/0.4537	<b>1.0000/1.0000</b>	0.6553/ <b>0.6242</b>	0.4873/0.4537	0.5634/0.5310
RoBERTa (large)	0.5027/0.4695	0.9221/0.9104	0.6506/0.6195	0.5174/0.4927	0.5013/0.4566
Our system (base)	0.4873/0.4537	<b>1.0000/1.0000</b>	0.6553/ <b>0.6242</b>	0.4873/0.4537	0.5484/0.4653
Our system (large)	0.4943/0.4574	0.9903/0.9032	<b>0.6595</b> /0.6072	0.5016/0.4699	<b>0.4794/0.4516</b>

Table 4: Performance of subtask 1c and 2a on the validation / test set.

and 16 multiheads. We used the Adam optimizer (Kingma and Ba, 2015) with learning rate  $5 \times 10^{-6}$ , and a batch size of 16. All the models were trained until the minimum loss value is reached on the validation set.

### 5.3 Evaluation Metrics

For classification tasks 1a and 1c, we use precision, recall, F-score, and accuracy as the evaluation metrics. For regression tasks 1b and 2a, we use the root mean square error as the evaluation metric:

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{n=1}^N (\hat{y}_n - y_n)^2}, \quad (5)$$

where  $\hat{y}_n$  is the predicted value, and  $y_n$  is the ground-truth value.

## 6 Results

The performance of our system and the baselines is shown in Table 3 (subtask 1a and 1b) and Table 4 (subtask 1c and 2a). We show the performance scores on both the validation and the test set. Generally speaking, the large version of our system performs quite well on all the four subtasks, compared with the other models. It can also be observed that, Transformer-based models always outperform the traditional methods by a large margin, except for subtask 1c, where all the models perform poorly and similarly. We conjecture this is because humor controversy is itself a highly subjective task, which is difficult even for humans. We also observe that large version of BERT-like models are generally better than their base counterparts, which is natural since larger models with more parameters usually bring better performance.

Table 5 gives the confusion matrix of our system on the test set in subtask 1a. We can see that in

		Ground-truth		Total
		P	N	
Predicted	P	582	24	606
	N	33	361	394
Total		615	385	1,000

Table 5: The confusion matrix of our system (large) on the test set (subtask 1a). P: Positive, N: Negative.

both positive and negative cases, the system performs quite well and makes only few errors. We manually examined some cases where our system makes a false prediction, and found that when our system predicts humorous but the ground-truth is non-humorous, the input text usually contains a question, e.g.,

*There are 2 kinds of families on Thanksgiving. Which one are you?*

We infer this is because most of the humorous examples in the training set contains a question, usually followed by a short answer serving as the punchline.

## 7 Conclusion

In this paper, we describe our system submitted to SemEval 2021 Task 7. We adopted the disentangled attention mechanism from the DeBERTa model, and participated in all the four subtasks. During the evaluation phase, we got a rank of 3 on the leaderboard for subtask 2a. For future work, we would like to combine human-centric features with the current architecture using the disentangled attention mechanism, and develop a hybrid model. In addition, we plan to expand the provided dataset with extra jokes from various sources such as Reddit forums, hoping to further improve the performance of our system.

## References

- Lei Chen and Chong Min Lee. 2017. [Convolutional neural network for humor recognition](#). *CoRR*, abs/1702.02584.
- Peng-Yu Chen and Von-Wun Soo. 2018. [Humor recognition using deep learning](#). In *Proceedings of NAACL-HLT 2018, Volume 2 (Short Papers)*, pages 113–117.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. [Transformer-XL: Attentive language models beyond a fixed-length context](#). In *Proceedings ACL 2019*, pages 2978–2988.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of NAACL-HLT 2019*, pages 4171–4186.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. [DeBERTa: Decoding-enhanced BERT with disentangled attention](#). *CoRR*, abs/2006.03654.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *Proceedings of ICLR 2015*.
- Herbert M Lefcourt and Rod A Martin. 2012. *Humor and life stress: Antidote to adversity*. Springer Science & Business Media.
- Lizhen Liu, Donghai Zhang, and Wei Song. 2018. [Modeling sentiment association in discourse for humor recognition](#). In *Proceedings of ACL 2018, Volume 2 (Short Papers)*, pages 586–591.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 task 7, HaHackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Rada Mihalcea and Carlo Strapparava. 2005. [Making computers laugh: Investigations in automatic humor recognition](#). In *Proceedings of HLT/EMNLP 2005*, pages 531–538.
- Rada Mihalcea, Carlo Strapparava, and Stephen G. Pulman. 2010. [Computational models for incongruity detection in humor](#). In *Proceedings of CICLing 2010*, volume 6008 of *Lecture Notes in Computer Science*, pages 364–374.
- Alex Morales and Chengxiang Zhai. 2017. [Identifying humor in reviews using background text sources](#). In *Proceedings of EMNLP 2017*, pages 492–501.
- Anton Nijholt, Oliviero Stock, Alan J. Dix, and John Morkes. 2003. [Humor modeling in the interface](#). In *CHI 2003 Extended Abstracts on Human Factors in Computing Systems*, pages 1050–1051.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of EMNLP 2014*, pages 1532–1543.
- Byron Reeves and Clifford Nass. 1996. *The media equation: How people treat computers, television, and new media like real people and places*. Cambridge University Press.
- Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. [Self-attention with relative position representations](#). In *Proceedings of NAACL-HLT 2018*, pages 464–468.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Proceedings of NeurIPS 2017*, pages 5998–6008.
- Orion Weller and Kevin D. Seppi. 2019. [Humor detection: A transformer gets the last laugh](#). In *Proceedings of EMNLP-IJCNLP 2019*, pages 3619–3623.
- Yubo Xie, Junze Li, and Pearl Pu. 2020. [Uncertainty and surprisal jointly deliver the punchline: Exploiting incongruity-based features for humor recognition](#). *CoRR*, abs/2012.12007.

# Grenzlinie at SemEval-2021 Task 7: Detecting and Rating Humor and Offense

**Ren yuan Liu**

Yunnan University, Yunnan, P.R. China  
bluwind159@qq.com

**Xiaobing Zhou \***

Yunnan University, Yunnan, P.R. China  
zhouxb@ynu.edu.com

## Abstract

This paper introduces the result of Team Grenzlinie's experiment in SemEval-2021 task 7: HaHackathon: Detecting and Rating Humor and Offense in English. This task has two sub-tasks. Subtask1 includes the humor detection task, the humor rating prediction task, and the humor controversy detection task. Subtask2 is an offensive rating prediction task. Detection task is a binary classification task, and the rating prediction task is a regression task between 0 to 5. 0 means the task is not humorous or not offensive, 5 means the task is very humorous or very offensive. For all the tasks, this paper chooses RoBERTa as the pre-trained model. In classification tasks, Bi-LSTM and adversarial training are adopted. In the regression task, the Bi-LSTM is also adopted. And then we propose a new approach named compare method. Finally, our system achieves an F1-score of 95.05% in the humor detection task, F1-score of 61.74% in the humor controversy detection task, 0.6143 RMSE in humor rating task, 0.4761 RMSE in the offensive rating task on the test datasets.

## 1 Introduction

Humorous is one kind most interesting, most has the power, most has the universal significance transmission art. Therefore, humor is one of the ways to improve the quality of daily conversation. In the field of natural language processing, how to make the computer learn humor and improve the quality of human-computer interaction is an important problem. The previous researches task was only to input the humorous corpus into the deep learning network and let the algorithm learn how to generate humorous dialogue. In this case, the sentences are often problematic. Because humor is an abstract concept, in different situations, the degree of humor and the way of humor will be different. Therefore, before the computer learns to generate humorous

sentences, it is an important task for the computer to understand humor and distinguish different degrees and forms of humor.

This paper mainly discusses how to identify these humorous sentences automatically. In SemEval-2021 task 7, subtask1 includes the humor detection task, the humor rating predicts task and the humor controversy detection task (Meaney et al., 2021). Subtask2 is an offensive rating predict task. In the detection task, the Bi-LSTM and adversarial training (Tramèr et al., 2017) is adopted, we also try to use FocalLoss to solve the data unbalance problem. In the regression task, the Bi-LSTM is also adopted. And then we propose a new method named compare method is also adopted.

The rest of the paper is as follows: Section 2 briefly introduces the related work. Section 3 describes the optimization approach to be used in detail. Section 4 describes the experiment process in detail. Section 5 is the conclusion of this paper.

## 2 Related Work

On large corpora, pre-trained models (PTMs) can learn common language representation, which is beneficial for subsequent NLP tasks and can avoid training new models from scratch (Wang et al., 2018). With the development of computing power and the improvement of training skills, the architecture of PTMs is advancing from shallow to deep.

The goal of the first version of PTMs is to learn good word embedding. Since these models are no longer needed by downstream tasks, they are usually very superficial for computational efficiencies, such as skip-gram (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Although these pre-trained embeddings can capture the semantic meaning of words, they are context-free and cannot capture the advanced concepts in the context, such as polysemy disambiguation, syntactic structure,

semantic role, anaphora, and so on. The second version of PTMs mainly learn context word embedding, such as ELMo(Peters et al., 2018), OpenAI, GPT(Radford et al., 2019) and BERT(Devlin et al., 2018). These learned coders still need to represent words in context through downstream tasks. Besides, various pre-training tasks are proposed to learn PTMs for different purposes.

Given the pre-trained model, downstream algorithms like Random Forest, SVM, Logistic Regression, or single linear layer can be adopted to get the result.

Recent studies have extended the humor detection task to the field of multi-modality and used gesture, speech prosody, and other features in humor detection task(Li et al., 2020). They also began to work on the joint training and integration of humor detection task and humor generation task(Weller et al., 2020). However, these models still regard humor detection as a binary classification task, without considering humor scoring and controversy.

### 3 Methods Description

**Baseline Model** In these tasks, RoBERTa is adopted as the baseline model, and softmax is adopted as the activation function in the classification task. In the regression task first, we try to use ReLU as the activation function, then we use sigmoid as the activation function, and multiply the output of sigmoid by 5. But we find the method without activation function does the best in regression task. If we add ReLU after regression task, the RMSE will reduce 0.1 0.2. So the activation function is not adopted in the baseline model. To avoid the negative output of the model in the scoring model, we need to add ReLU as an activation function in the test phase.

In the regression task, the Mean Square Loss is adopted as the loss function. In classification tasks, we use the CrossEntropy Loss as the loss function.

**Method1: Bi-LSTM** In this paper, all the sub-task use the Bi-LSTM to extract more abundant features. In this model, Bi-LSTM is added after the pre-trained model. [CLS] (classification symbol) always be added before sentence, and use classifier to compute [CLS] representation to get the result. So, there is a problem, that is, the sentence representation from Bi-LSTM will not integrate on symbol [CLS]. But we need the representation of [CLS] for the next step. So the output of Bi-LSTM is sent into a new defined transformer layer, encode

the sentence representation into the symbol [CLS]. Finally, the sentence representation will send into a single linear layer to get the result.

**Method2: adversarial training** Then the adversarial training is adopted to improve the baseline model. Adversarial training is an important way to enhance the robustness of neural networks. In the process of confrontation training, the samples will be mixed with some small disturbances, and then make the neural network adapts to this change, so it has the robustness to the confrontation samples. In the field of the language model, adversarial training improves both robustness and generalization (Morris et al., 2020).

Adversarial training can be summarized as the following maximum and minimum formula,

$$\min_{\theta} \mathbb{E}_{(Z,y) \sim \mathcal{D}} \left[ \max_{\|\theta\| \leq \epsilon} (L(F_{\theta}(X + \delta)), y) \right] \quad (1)$$

Where  $X$  represents the input representation of the sample,  $\theta$  represents the disturbance superimposed on the input,  $F_{\theta}()$  is the neural network function,  $y$  is the label of the sample, and  $L(F_{\theta}(X + \delta)), y$  represents the loss obtained by superimposing a disturbance  $\theta$  on the sample  $X$ , and then comparing it with the label  $y$  through the neural network function.  $\max(L)$  is the optimization objective, that is to find the disturbance that maximizes the loss function. In short, the added disturbance should confuse the neural network as much as possible.

$\min_{\theta} \mathbb{E}_{(Z,y) \sim \mathcal{D}}$  is the minimization formula to optimize the neural network, that is, when the disturbance is fixed, we train the neural network model to minimize the loss of training data, that is to say, the model has certain robustness and can adapt to the disturbance.

In this method, FGM (Fast Gradient Method) (Miyato et al., 2016) is adopted. The idea is very simple, that is, let the direction of disturbance increase along the gradient, and the increase along the gradient means the maximum loss. The formula of FGM is as follows.

$$\delta = \epsilon \cdot \frac{g}{\|g\|_2} \quad (2)$$

Where  $\epsilon$  is a constant, which controls the degree of disturbance rejection.  $g = \nabla X(L(F_{\theta}(X)), y)$ , i.e. the gradient of loss function  $L$  with respect to input  $X$ .

**Method3: FocalLoss** In the classification task, we can see that the data ratio of humorous and non-humorous sentences is close to 2:1, and then in humor’s data, the ratio of controversy and non-controversy is close to 1:1, but We assume that non-humorous sentences are also non-controversy sentences. Therefore, these two tasks are faced with the problem of data imbalance. To solve this problem, we use the FocalLoss (Lin et al., 2017) as the loss function. The FocalLoss is as follows.

$$FocalLoss(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \quad (3)$$

Where  $p_t$  is the probability of the label  $t$  that is outputted by the classifier.  $N$  is the number of labels.  $\alpha$  and  $\gamma$  are constant.

**Method4: Compare Method** This method is only for humor rating predict task and offensive rating predict task. In the few-shot classification tasks, traditional approach is given a pair of sentences in the same class and given a pair of sentences in different classes. Let the classifier identify whether the pair of sentences is the same class or different classes. So the latent feature of each label in the sentence can be extracted. Unfortunately, this approach can’t be used in regression tasks.

Based on this idea, we proposed the compare method. This method extends the above idea to the regression task. The approach is shown in figure 1. In this model, we input sentence A with rating  $L(A)$ , and sentence B with rating  $L(B)$ , then three different models that realize the function of  $M_{add}$  ( $L(A)$  plus  $L(B)$ ),  $M_{sub\_AB}$  ( $L(A)$  minus  $L(B)$ ), and  $M_{sub\_BA}$  ( $L(B)$  minus  $L(A)$ ). It means to use these models to encode pairs of sentences and output  $Z_{add}$ ,  $Z_{sub\_BA}$ ,  $Z_{sub\_AB}$ . Then put these features into the classifier. Let the output ratings become the addition and the subtraction of the pair of sentences’ rating.

Furthermore, the sentence representation which rating is close to the addition and subtraction of the pair of sentences’ ratings can be used to introduce the  $Z_{add}$ ,  $Z_{sub\_BA}$ ,  $Z_{sub\_AB}$  by minimizing the MSELoss of  $Z_{add}$ ,  $Z_{sub\_BA}$ ,  $Z_{sub\_AB}$  and sentence representation. In this task, this approach is not adopted because of the lack of data.

The three models have the same construction. To simplify the computation, the last layer’s hidden output from RoBERTa is set as the feature of each token. Then these token features are concatenated like "[CLS] (sentence) [SEP] (another sentence) [SEP]". And send the concatenated output to a

single transformer layer to get the [CLS] output for classifying.

The loss function  $C_{loss}$  and the  $Add_{loss}$  is as follows. In the equation, ML is MSELoss and C is our model,  $C(F_A)$  means the output of our model. These loss functions are the loss of the single sentence result and the loss of the result of  $Z_{add}$ ,  $Z_{sub\_BA}$ ,  $Z_{sub\_AB}$ .

$$C_{loss} = ML(C(F_A), L(A)) + ML(C(F_B), L(B)) \quad (4)$$

$$Add_{loss} = ML(C(Z_{add}), L(A) + L(B)) + ML(C(Z_{sub\_AB}), L(A) - L(B)) + ML(C(Z_{sub\_BA}), L(B) - L(A)) \quad (5)$$

## 4 Experiment Setup

**Datasets** First of all, we try to find the relationship between tasks. In the beginning, we think that those with low humor ratings or high offensive ratings may be controversial, but unfortunately, we find many Counterexamples in the datasets. Then we tried to train several tasks together, but the result was not as good as that of training it independently. So we train these tasks independently.

Secondly, in the task of humor scoring and humor controversy detection, only humorous sentences need to be rating predicted and detected. In the data set, only humorous sentences have humor ratings and humor controversy labels. Therefore, how to deal with the label of non-humorous sentences is an important problem. We have tried to set the controversy label of non-humorous sentences to 2, that is, the third category, but this approach will identify humorous sentences as the third category, which will interfere with the model. Therefore, in this paper, we set the rating of non-humorous sentences to 0, and the controversy label to 0, i.e. non-controversy.

**Parameters setting** In this section, the hyperparameter is the same in all subtasks. The optimizer is AdamW with a 3e-5 learning rate and 1e-8 adam epsilon. The pre-trained model has 12 transformer layers and 768 hidden sizes. The max sequence length is 180. The batch size is 8. And weight decay is 0.

## 5 Result

The result of the test datasets is shown in Table 1. Final results in line 1 is results in evaluation

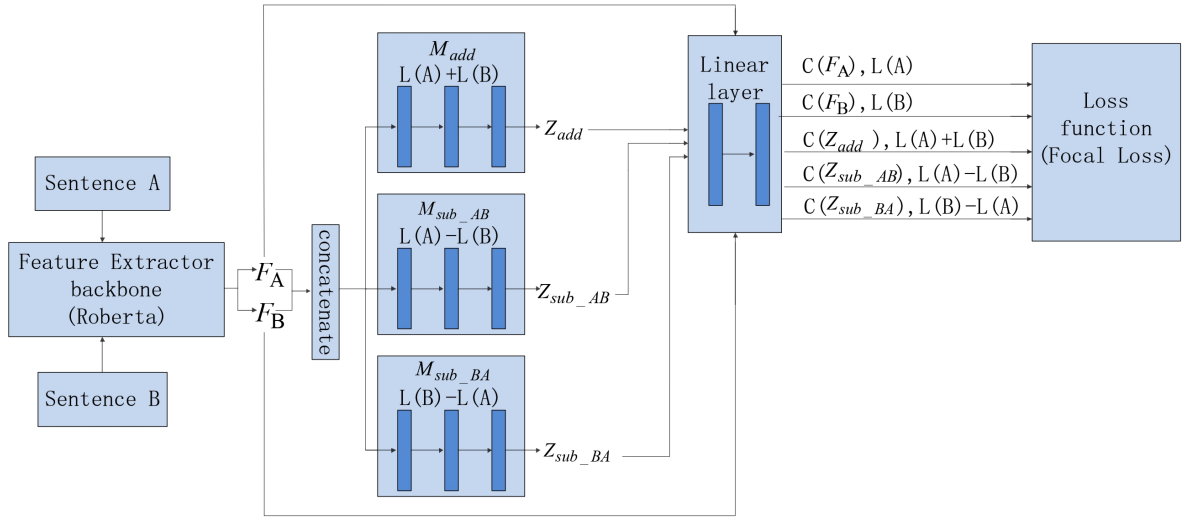


Figure 1: Compare Method Construction

phase. FGM+FocalLoss and compared method are adopted. and other line are the result in post evaluation phase. In this phase, we reduce the learning rate. In table 1, it can be seen that the result of all optimize methods in the controversy detection task is worse than the baseline model. Because we set the non-humorous sentences as non-controversy, This will greatly interfere with the model’s judgment of non-controversy sentences. In the evaluation datasets, i.e. predicted results from non-humorous sentences are used to calculate the F1-score, these approaches do optimize the baseline model. But these approaches do not play an optimization role in the test phase. So we can make this conclusion. Then FGM and Bi-LSTM will make the model extract more abundant features, which will undoubtedly aggravate the interference of non-humorous sentences and reduce the prediction accuracy of the model.

FocalLoss didn’t work as expected and didn’t get better results. Because FocalLoss usually use in the datasets that 0 label is more than 1 label, but in the humor detection task, 1 label is more than 0. Although we adjusted the alpha in FocalLoss to 0.67, FocalLoss still failed to get better results.

FGM optimizes the baseline model in humor detection, humor rating, and offensive rating tasks. and based on FGM, Bi-LSTM does more better in these tasks. Because Bi-LSTM can extract sentence features in more detail, especially bidirectional sequence features. Experiments show that these features are more conducive to downstream tasks.

Finally, Compare Method only optimizes the

offensive rating predict task, but it not good at humor rating predict task, we think the non-humorous sentences. We speculate that non-humorous sentences with a 0 rating interferes with the comparison of two randomly selected sentences in compare method. The number of sentences that select non-humorous sentences for comparison is too large to help the model predict rating, so the auxiliary task interferes with the baseline model.

## 6 Conclusion

This paper introduces the experiment in SemEval-2021 task 7 HaHackathon: Detecting and Rating Humor and Offense. In this article, we propose two main assumptions. The first point is that the model is difficult to obtain the real meaning of the tag according to the change of the 0-5 rating. So the method of adding the auxiliary task on the baseline model was proposed. The auxiliary task is comparing different sentences according to the number proposed by us all to strengthen and supplement this process. This method does the best in offensive rating predict task, achieve 0.4761 RMSE. Second, the output of the pre-training model is similar to the word vector, which needs further processing to be more suitable for downstream tasks. So we try to use Bi-LSTM. Indeed Bi-LSTM does the best, achieve the 95.05% F1-Score in the humor detection task, and 0.6143 RMSE in the humor rating task. These approaches do not play an optimized role in the controversy detection task. The baseline does the best, achieve the 61.74% F1-score. The main reason for this problem lies in the interference



Model	Humor F1	Humor RMSE	Controversy F1	Offensive RMSE
Final results	0.9386	0.6312	0.5455	0.4761 RoBERTa
0.9344	0.6961	0.6174	0.5146	
FGM	0.9481	0.6311	0.5614	0.4847
Bi-LSTM+FGM	0.9505	0.6143	0.5609	0.4956
FGM+FocalLoss	0.9386	-	0.5454	-
Compare Method	-	0.6906	-	0.4761

Table 1: The result of several optimize approach on test datasets

of non-humorous sentences. So there is still room for improvement, such as eliminating the influence of non-humorous sentences, adjust the model parameters and try other pre-trained models. Or try to use a classification model and regression model in machine learning, such as Bayesian or CRF, to process the output of BERT. Therefore, the future work is to find a better way to remove the influence of non-humorous sentences and find a better way to optimize the controversy detection task. And then do more experiments to get better results.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Lily Li, Or Levi, Pedram Hosseini, and David A. Bro-niatowski. 2020. A multi-modal method for satire detection using textual and visual cues.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.
- Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations*.
- J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- F Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. 2017. Ensemble adversarial training: attacks and defenses.
- Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322:93–101.
- Orion Weller, Nancy Fulda, and Kevin Seppi. 2020. Can humor prediction datasets be used for humor generation? humorous headline generation via style transfer. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 186–191, Online. Association for Computational Linguistics.

# abcbpc at SemEval-2021 Task 7: ERNIE-based Multi-task Model for Detecting and Rating Humor and Offense

Chao Pang, Xiaoran Fan, Weiyue Su, Xuyi Chen  
Shuohuan Wang, Jiayang Liu, Xuan Ouyang  
Shikun Feng, Yu Sun

Baidu Inc., China

{pangchao04, fanxiaoran, suweiyue, chenxuyi}@baidu.com  
{wangshuohuan, liujiayang, ouyangxuan}@baidu.com  
{fengshikun01, sunyu02}@baidu.com

## Abstract

This paper describes our system participated in Task 7 of SemEval-2021: Detecting and Rating Humor and Offense. The task is designed to detect and score humor and offense which are influenced by subjective factors. In order to obtain semantic information from a large amount of unlabeled data, we applied unsupervised pre-trained language models. By conducting research and experiments, we found that the ERNIE 2.0 and DeBERTa pre-trained models achieved impressive performance in various subtasks. Therefore, we applied the above pre-trained models to fine-tune the downstream neural network. In the process of fine-tuning the model, we adopted multi-task training strategy and ensemble learning method. Based on the above strategy and method, we achieved RMSE of 0.4959 for sub-task 1b, and finally won the first place.

## 1 Introduction

Humor, as a highly subjective phenomenon, can be affected by various factors. Automatic humor recognition relies on annotated data to determine whether the text is humorous or not (Mihalcea and Strapparava, 2005). However, such a binary classification does not capture the level of humor, so assessing the level of humor is of great significance (Garimella et al., 2020). Since humor can be influenced by many factors, such as age, and may offend others. Based on such a situation, SemEval-2021 Task 7 focuses on linking humor and offense across different age groups (Meaney et al., 2021). But there are still many challenges to this task. For example, the dataset for the task is small and the texts are short, which does not allow for adequate training. To address these issues, we utilized unsupervised pre-trained language models and fine-tuned these models for specific downstream subtasks. After conducting research and extensive comparative

experiments, the results show that ERNIE 2.0 (Sun et al., 2019b) and DeBERTa pre-trained models performed best on the subtasks. These large unsupervised language models were pre-trained on a large amount of unlabeled data to extract valuable lexical, syntactic, and semantic information from the corpus. Vector representations of text computed by these models are applied to fine-tune the downstream neural networks for the subtasks. The multi-task training and ensemble learning method significantly improve our model’s performance.

The rest of the paper is organized as follows: Section 2 provides a brief description of the related work, and Section 3 describes our proposed approach in detail. In Section 4, the experiments are described in detail and the results are presented. Finally, we summarize the whole paper and discuss future research directions in Section 5.

## 2 Related Work

In the early research of humor and offense detection and evaluation, traditional machine learning methods and n-gram language model were mostly used.

Recent research has shown that unsupervised language pre-trained models using large amounts of unlabeled data have achieved state-of-the-art results in a large number of natural language processing tasks. For example, BERT (Devlin et al., 2018) is a model built based on Transformer Encoder, which is used for downstream tasks by pre-training on the masked language models task and the next sentence prediction task, and then for fine-tuning. Inspired by this approach, many pre-training language models have been proposed. For example, ALBERT (Lan et al., 2019) adopts Factorized Embedding Parameterization and Cross-layer parameter sharing strategies, and adds the sentence order prediction task, so that the model can greatly

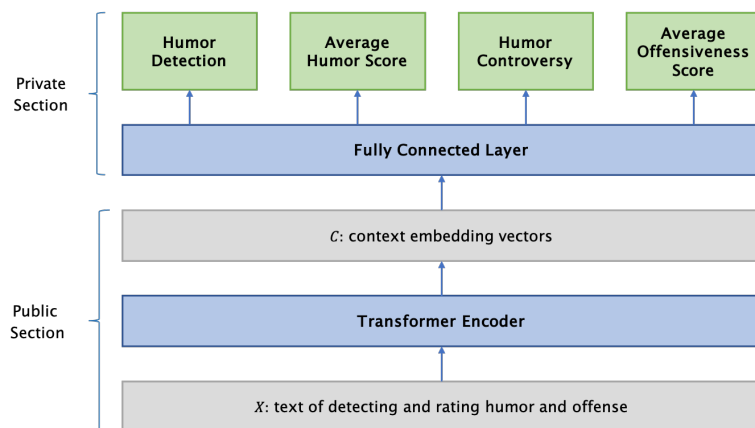


Figure 1: Architecture of our model for multi-task training. The public section is shared by all subtasks, while the private section is subtask-specific. First, text is transformed into subwords, and ERNIE 2.0 generates the corresponding contextual vector representation. Then different loss functions are set for each subtask to generate task-specific representations. This is eventually used for classification and regression subtasks in different scenarios.

reduce the number of parameters without lose accuracy compared to BERT. RoBERTa (Liu et al., 2019b) removes the next sentence prediction task and uses dynamic MASK, which is optimized for BERT. DeBERTa (He et al., 2020) improves the BERT and RoBERTa models by using two new techniques. The first one is using Disentangled attention mechanism and secondly, Enhanced mask decoder is used. MT-DNN (Liu et al., 2019a) combines Multi-task Learning and pre-trained models to improve the performance of various natural language processing tasks.

ERNIE 1.0 (Sun et al., 2019a) employs an entity-level and phrase-level mask, the extension of its training corpus and the use of multiple rounds of conversation to replace sentence pair classification further enhance the model’s semantic representation capability. ERNIE 2.0 (Sun et al., 2019b) is an optimized version of ERNIE 1.0, which introduces a large number of pre-training tasks and continuously updates the pre-training model through multi-task learning to help the model learn lexical, syntactic and semantic representations efficiently. ERNIE 2.0 constructs three pre-training tasks, namely word-aware pre-training tasks, structure-aware pre-training tasks and semantic-aware pre-training tasks. The performance of the model is improved by constructing pre-training tasks from multiple perspectives. The ERNIE 2.0 model outperformed BERT and XLNet (Yang et al., 2019) almost across the board on the English task and achieved the best results on 7 GLUE tasks; on the Chinese task, the ERNIE 2.0 model outperformed

BERT across the board on all 9 Chinese NLP tasks.

### 3 Our Approach

#### 3.1 Multi-task training

To mitigate overfitting for specific tasks, we adopt a multi-task training strategy that combines pre-trained language model and multi-task training (as shown in Figure 1). Based on the above strategy, it makes the learned representation generalizable across tasks and improves the performance of various downstream subtasks.

The architecture of our model is shown in Figure 1, with the pre-trained model ERNIE 2.0 as the public section, which is used for generating semantic information common to downstream tasks. and the multi-task training as the private section, where individual subtasks are trained to produce task-specific representations by using different loss functions.

When we fine-tune our model, the input to the model is the data from all subtasks. Words from the text in different subtasks are first processed by tokenizer to generate subwords. After the subwords are transformed into tokens by the mapping of the lexicon, the tokenized sentences are stitched together with [CLS] and [SEP] as the input to the ERNIE 2.0 model to obtain the contextual vector representation corresponding to each token. Multi-task training is a fully-connected layer followed by the ERNIE 2.0 model, and the four downstream subtasks optimize the subtask-specific model by constructing different loss functions for gradient

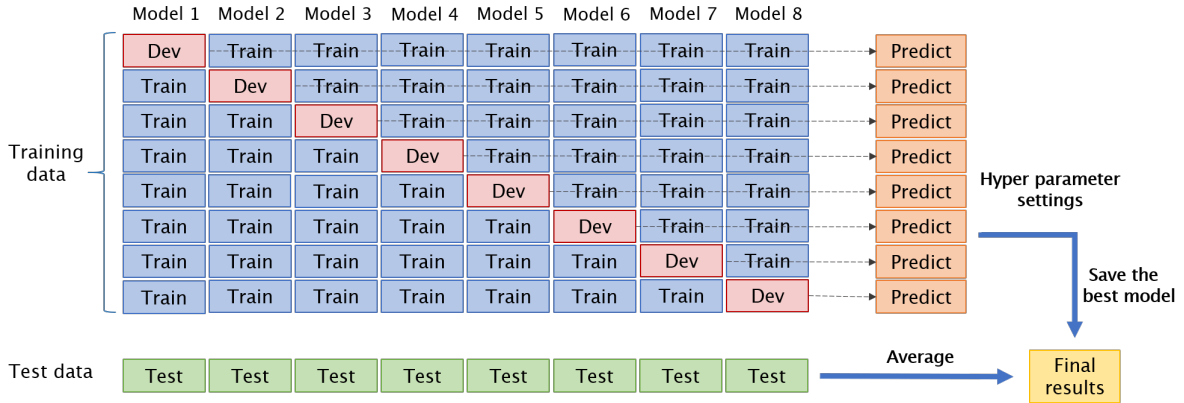


Figure 2: 8-fold cross-validation and ensemble. The training set is divided randomly for 8 times by setting different random seeds. In each division, the training set  $T$  is divided into 8 parts, of which 7 parts are respectively used as the training set and the remaining 1 part is used as the validation set. And finally the average of all saved best models predicted on the test set are the final results.

updating. For the classification task we first use the sigmoid activation function to constrain the output between 0 and 1 before using the BCE loss function, and for the regression task we use the MSE loss function.

### 3.2 Ensemble

We adopt cross-validation for training as a way to improve the robustness of our model, as shown in Figure 2. We first divided the training set eight times by setting different random seeds. Therefore, 8 folds of data are generated, with 7000 training samples and 1000 validation samples in each fold. When fine-tune our model for each fold, the best model for each subtask at each fold of training is saved. For subtask 1a and subtask 1c, the evaluation metric is F1-Score, and for subtask 1b and subtask 2a the evaluation metric is RMSE. finally, we take the mean of all the best saved models after making predictions on the test set as the final results.

## 4 Experiment

### 4.1 Experimental details

All of our experiments were run on the Nvidia Tesla V100. In order to obtain more valuable information from the limited training data and to reduce overfitting to some extent, we adopt the multi-task training strategy and ensemble learning method.

For the training of the per-fold model, we choose the Adam optimizer, set the epoch to 10, and use early stopping strategies according to the performance on the validation set. Considering the small amount of data in the training set, we set a smaller

learning rate for the ERNIE 2.0 model layer and a larger learning rate for the fully connected layer where the subtasks are trained together. Specifically, when fine-tuning our model, we adopt the grid search strategy with the learning rate ranging from  $2e-5$  to  $5e-5$  and the batch size ranging from 32 to 48. Besides, we set the learning rate as a linear function and use a warm-up strategy in the training phase. The ensemble approach we adopt is mainly based on the average prediction results. The specific methods are as follows: for the classification subtask, the prediction probabilities of all base models in each category are averaged, and then the category with the highest probability is taken as the prediction result; while for the regression subtask, the prediction values of all base models are averaged as the final prediction result.

### 4.2 Comparison experiments

In order to verify the effectiveness of our proposed multi-task training strategy based on the ERNIE 2.0 model, we set up two comparison experiments. They are described as follows:

(1) Comparison experiments of multi-task training together and single-task training separately based on ERNIE 2.0 model.

(2) Comparison experiments of single-task separate training based on ERNIE 2.0 and DeBERTa models.

### 4.3 Experimental Results

Table 1 summarizes the results on the validation set of all the models we tried based on the 8-fold cross-validation method. We can see that under

Models	Is_multi_task	Task 1a (F1-Score)	Task 1b (RMSE)	Task 1c (F1-Score)	Task 2a (RMSE)
DeBERTa <sub>xlarge</sub>	False	0.9603	0.4492	0.6598	0.4817
ERNIE 2.0 <sub>xlarge</sub>	False	0.9656	0.4542	0.6388	0.4746
ERNIE 2.0 <sub>xlarge</sub>	True	0.9727	0.4475	0.6566	0.4722

Table 1: The results of different models under the 8-fold cross-validation method. The table describes the results of the DeBERTa and ERNIE 2.0 models on four subtasks in the case of single-task training separately and multi-task training together. For subtask 1a and subtask 1c, the evaluation metric is F1-Score, and for subtask 1b and subtask 2a the evaluation metric is RMSE.

the evaluation metrics of each subtask, The number of parameters in the ERNIE 2.0 model (425M) is less than the DeBERTa model (750M), but the ERNIE 2.0 pre-trained model performs better than the DeBERTa model on several subtasks. Besides, Compared to single-task training, the strategy of using multi-task training shows a significant improvement in performance. Moreover, The impact of ensemble learning method on improving model performance is significant.

## 5 Conclusion

In this paper, we proposed a multi-task training system based on ERNIE 2.0. We describe the architecture of the model and the training process in detail. Besides, we experimentally demonstrate that the strategy performs better with multi-task training compared to single-task training. Moreover, the ensemble learning method makes the model more robust. As a result, we have won the first place in a subtask for the competition of SemEval-2021 task 7. In our future work, we will further explore pre-trained language model and optimize the multi-task training.

## References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional Transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Aparna Garimella, Carmen Banea, Nabil Hossain, and Rada Mihalcea. 2020. “judge me by my size (noun), do you?” YodaLib: A demographic-aware humor generation framework. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2814–2825, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Rada Mihalcea and Carlo Strapparava. 2005. Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019a. ERNIE: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019b. ERNIE 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

# Humor@IITK at SemEval-2021 Task 7: Large Language Models for Quantifying Humor and Offensiveness

Aishwarya Gupta\*, Avik Pal\*, Bholeshwar Khurana\*, Lakshay Tyagi\*,  
Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{aishwaryag20, avikpal, bholek, lakshayt}@iitk.ac.in  
ashutoshm@cse.iitk.ac.in

## Abstract

Humor and Offense are highly subjective due to multiple word senses, cultural knowledge, and pragmatic competence. Hence, accurately detecting humorous and offensive texts has several compelling use cases in Recommendation Systems and Personalized Content Moderation. However, due to the lack of an extensive labeled dataset, most prior works in this domain haven't explored large neural models for subjective humor understanding. This paper explores whether large neural models and their ensembles can capture the intricacies associated with humor/offense detection and rating. Our experiments on the SemEval-2021 Task 7: HaHackathon show that we can develop reasonable humor and offense detection systems with such models. Our models are ranked third in subtask 1b and consistently ranked around the top 33% of the leaderboard for the remaining subtasks.

## 1 Introduction

Like most figurative languages, humor/offense pose interesting linguistic challenges to Natural Language Processing due to its emphasis on multiple word senses, cultural knowledge, sarcasm, and pragmatic competence. A joke's perception is highly subjective, and age, gender, and socioeconomic status extensively influence it. Prior humor detection/rating challenges treated humor as an objective concept. SemEval 2021 Task 7 (Meaney et al., 2021) is the first humor detection challenge that incorporates the subjectivity associated with humor and offense across different demographic groups. Users from varied age groups and genders annotated the data with the text's humor and have provided an associated score for the same. It is also quite a generic phenomenon that a text might be

\* Authors contributed equally to the work. Names in alphabetical order.

humorous to one and normal/offensive to another. Rarely has it been noticed that the same content is globally accepted as witty. To the best of our knowledge, Meaney et al. (2021) is the first initiative towards annotating the underlying humor as controversial or not. Understanding whether a text is humorous and/or offensive will aid downstream tasks, such as personalized content moderation, recommendation systems, and flagging offensive content.

Large Language Models (LLMs) have recently emerged as the SOTA for various Natural Language Understanding Tasks (Lewis et al., 2019; Raffel et al., 2019; Conneau et al., 2019; Zhang et al., 2020). However, typical day-to-day texts, where these models have shown state of the art performance, are less ambiguous than texts having puns/jokes. Training and evaluating LLMs in the context of highly ambiguous/subjective English texts would serve as an excellent benchmark to figure out the current shortcomings of these models. This paper studies various large language models – BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019), ERNIE-2.0 (Sun et al., 2019) and DeBERTa (He et al., 2020) and their ensembles – for humor and offense detection tasks. Additionally, we explore a Multi-Task Learning framework to train on all the four sub-tasks jointly and observe that joint training improves the performance in regression tasks.

We have achieved significant performance on all the subtasks and have consistently ranked  $\sim \frac{1}{3}$ <sup>rd</sup> of the total submissions. We were ranked (1) 21<sup>st</sup> with an F-score and accuracy of 94.8% and 95.81% respectively in Task 1a, (2) 3<sup>rd</sup> with an RMSE score of 0.521 in Task 1b, (3) 9<sup>th</sup> with an F-score and accuracy of 45.2% and 62.09% respectively in Task 1c; and (4) 16<sup>th</sup> with an RMSE score of 0.4607 in Task 2. We release the code for models

and experiments via GitHub<sup>1</sup>

We organize the rest of the paper as: we begin with a description of the challenge tasks followed by a brief literature survey in section 2. We then describe all of our proposed models in section 3 with training details in section 4 and present the experimental results in section 5. Finally, we analyze our findings and conclude in section 6, and 7 respectively.

## 2 Background

### 2.1 Problem Description

SemEval 2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense (Meaney et al., 2021) involves two main tasks – humor detection and offense detection. The organizers further subdivide the task into following subtasks:

1. Humor detection tasks:
  - (a) **Task 1a** involves predicting whether a given text is humorous.
  - (b) **Task 1b** requires predicting the humor rating of a given humorous text.
  - (c) **Task 1c** incorporates humor subjectivity by posing a classification problem of predicting whether the underlying humor is controversial or not.
2. **Task 2** is an offense detection task and is posed as a bounded regression problem. Given a text, we need to predict a mean score denoting the text’s offensiveness on a scale of 0 to 5, with 5 being the most offensive.

### 2.2 Related Works

**Transfer Learning** ULMFiT (Howard and Ruder, 2018) used a novel neural network based method for transfer learning and achieved SOTA results on a small dataset. Devlin et al. (2018) introduced BERT to learn latent representations in an unsupervised manner, which can then be finetuned on downstream tasks to achieve SOTA results. Lan et al. (2019); Liu et al. (2019); Sanh et al. (2019); Sun et al. (2019) have proposed several improvements to the BERT model. In this paper, we analyze the effects of using these different base models in the context of humor and offense detection.

<sup>1</sup><https://github.com/aishgupta/Quantifying-Humor-Offensiveness>

**Humor & Emotion Detection** Weller and Seppi (2019) first proposed the use of transformers (Vaswani et al., 2017) in humor detection and outperformed the state of the art models on multiple datasets. Ismailov (2019); Annamradnejad (2020) extended the use of BERT models to humor classification. Fleşcan-Lovin-Arseni et al. (2017) did humor classification by comparing and ranking tweets while Docekal et al. (2020) edit the tweet and rank the extent of humor for the edited tweet on a scale of 0 to 3 (most funny). There has been extensive research in the area of text emotion prediction and generation (e.g., Witon et al. (2018); Colombo et al. (2019); Goswamy et al. (2020); Singh et al. (2021)). Demszky et al. (2020) curated a large scale emotion detection dataset and achieved SOTA results by finetuning a BERT model. However, none of these works delve into humor analysis’ subjectivity, which is a prime focus of this task.

**Sentiment and Pun Analysis** Li et al. (2019); Mal-toudoglou et al. (2020) study BERT based models for sentiment analysis. Ke et al. (2019) uses a combination of sentence embedding, POS tagging and word-level sentiment polarity scores for sentiment classification. Zhou et al. (2020) uses contextualized and pronunciation embeddings for each word and pass these through a neural network to detect and localize pun in the sentence. However, none of these works focus on the subjectivity of the underlying sentiment and pun in the text.

## 3 System Overview

### 3.1 Data

The challenge dataset comprises of a `train` set (labeled 8000 texts) and a `public-dev` set (labeled 1000 texts). Each text input is labeled as 1/0 if it is humorous or not and rated with the offensiveness score on a scale of 0-5. If a text is classified as humorous, it is further annotated with humor rating and classified as controversial or not. For our single-task models (Section 3.2), we train on the `train + public-dev` set after obtaining a suitable stopping epoch by training and validating on the `train` and `public-dev` respectively. For our multi-task models (Section 3.3), we train on 8200 texts sampled randomly from `train` and `public-dev` sets and use remaining 800 text inputs for validation.

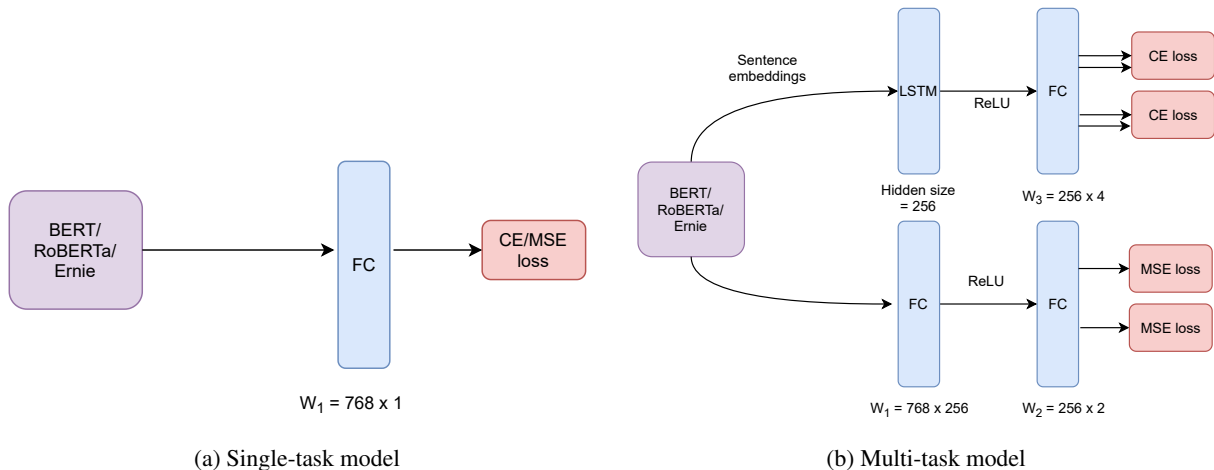


Figure 1: Different Model architectures used for Humor/Offense detection/rating.

### 3.2 Single Task Model

As the tasks are evaluated independently, we have explored LLMs for each task/subtask independently and will be referring to them as single task models. Inspired by [Demszky et al. \(2020\)](#), for each task, we add a classification (for Task 1a, 1c) or a regression (for Task 1b, 2) head on top of the pretrained models like BERT, RoBERTa, ERNIE-2.0, DeBERTa and XLNet and train the model end-to-end (Figure 1a). This ensures that the model learns features solely related to the task, enhancing the performance. Also, as we only add a classification/regression head, the number of learnable parameters does not increase much. This helps us in finetuning the model on such a small dataset for a few number of epochs avoiding overfitting and resulting in better generalization.

### 3.3 Multi Task Learning

[Collobert and Weston \(2008\)](#) demonstrated that Multi-Task Learning (MTL) improves generalization performance across tasks in NLP. The different tasks though uncorrelated, share the same underlying data distribution. This can be of great help for tasks 1b and 1c where labeled instances are far less than for task 1a or 2. Exploiting the fact that all tasks share same data distribution, we propose to learn a model jointly on all the tasks. Specifically, we consider hard parameter sharing among different tasks and parameterize the base models using a neural network, followed by two heads for classification and regression tasks (Figure 1b). Our base model includes LLMs like BERT, RoBERTa, and ERNIE. Contrary to the LSTM layer, which helps in learning features using all the token level embed-

dings, the Fully Connected (FC) layer focuses only on the embedding of [CLS] token. Hence, having these two branches allow the model to focus on different tasks using the same sentence embedding and helps in learning enhanced embeddings for task 1b and 1c with much lesser labeled dataset.

### 3.4 Ensembles

Mostly LLMs differ in their training procedure, and architecture. These big language model frameworks are trained on wide set of datasets for a variety of tasks. Though, they all have comparable performance, they may still capture different aspects of the input. We try to leverage such varied informative embeddings based predictions by combining multiple models trained with different basenet using following strategies:

**Jointly trained Model Embeddings:** All the big language frameworks have shown huge performance improvement on multiple tasks owing to their highly informative latent input embeddings. We propose to learn an ensemble leveraging diverse aspects of the input captured by varied LLMs by concatenating their latent embeddings and mapping them to low dimensional space for task prediction. We use this method in learning ensembles of single task models explained in 3.2.

**Aggregation of Trained Model Predictions:** Joint-training though more informative and powerful, is a computationally intensive approach. Thus as an alternative, we use a weighted averaging of multiple pretrained models without compromising much on the performance.

1. **Weighted Aggregate of Regression Outputs:** For an ensemble of  $k$  models trained



Model	Task1-a		Task1-b	Task1-c		Task2
	F-Score	Accuracy	RMSE	F-Score	Accuracy	RMSE
STM (BERT)	-	-	0.5841	0.5934	0.4829	0.4997
STM (RoBERTa)	0.9523	0.9410	0.5929	<b>0.6242</b>	0.4536	-
STM (ERNIE-2.0)	0.9541	0.9430	0.5546	0.4113	0.5252	0.4716
STM (XLNet)	-	-	0.5656	0.5892	0.5171	-
STM (DeBERTa)	0.9532	0.9420	0.5491	-	-	-
STM (Agg. Ensemble)	<b>0.9581</b>	<b>0.9480</b>	0.5480	0.4520	<b>0.6209</b>	0.4750
MTM (BERT)	0.9374	0.9210	0.5794	0.5080	0.5496	0.5049
MTM (RoBERTa)	0.9477	0.9350	0.5873	0.5479	0.5170	0.5141
MTM (ERNIE-2.0)	0.9530	0.9420	0.5541	0.5389	0.5187	0.4961
STM + MTM (Agg. Ensemble)	0.9520	0.9400	<b>0.5210</b>	0.5321	0.5252	<b>0.4520</b>

Table 1: Metrics on the test dataset for the major models on all the sub-tasks. MTM stands for Multi-Task Model, STM stands for Single Task Model, and Agg. Ensemble is Aggregation Based Ensembling without having to jointly train all the models together.

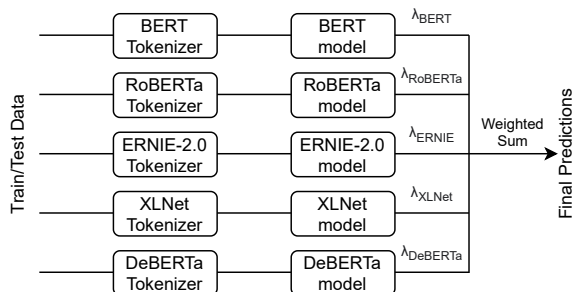


Figure 2: **Weighted-Average Ensembling:** The data is tokenized and then passed to the respective model. A weighted sum is done to obtain the final predictions.  $\lambda_i$  represents the weight for model  $i$ .

using different LLMs as basenet, the aggregate output  $\hat{y}$  is computed as  $\hat{y} = \sum_{i=1}^k \lambda_i \cdot \hat{y}_i$  where  $y_i$  and  $\lambda_i$  represents the output and weight of the  $i^{th}$  model respectively. The weights  $\lambda_i$  are obtained through extensive grid search on the held out validation dataset or set to a  $\frac{1}{k}$  when trained on the entire dataset without a validation set. The complete approach is shown in figure 2.

2. **Voting Based Classification:** This is one of the most popular approach of learning an ensemble and does not involve any hyperparameters or retraining of any of the constituent models. This involves training multiple models independently and using maximum among all the predictions as the final output. For a binary classification task, the final output  $\hat{y}$  is by max-voting across the independent models.

## 4 Experimental Setup

We used Pytorch (Paszke et al., 2019) and HuggingFace (Wolf et al., 2020) library for our models, and Google Colab GPUs for training and inference. We use ADAMW (Loshchilov and Hutter, 2019) and ADAM (Kingma and Ba, 2017) optimizer with initial learning rate of  $2e^{-5}$  for training single task and multi task models respectively. For each of the models we follow a dedicated training pipeline described in subsequent sections.

### 4.1 Data preprocessing

We split the dataset into training and validation data as described in Section 3.1. The sentences are annotated with a [CLS] token in the beginning and given as an input to the model. We performed additional experiments by removing stopwords but noticed a slight deterioration in the performance.

### 4.2 Loss Functions

Task 1a & 1c are instances of binary classification problem and thus have been trained using cross-entropy loss. For predicting humor and offense rating i.e., Task 1b and 2, we have used mean squared error as the loss function.

### 4.3 Training Details

All the models are trained for  $n$  epochs where  $n$  is a hyper-parameter tuned on the validation set using early stopping criteria. For single task models, we split train data into training and validation set to learn the optimal value of  $n$  and then train the model from scratch on train + public-dev

Rank	Task1-a		Task1-b RMSE	Task1-c		Task2 RMSE
	F-Score	Accuracy		F-Score	Accuracy	
Rank-1	0.982	0.9854	0.4959	0.4943	0.6302	0.4120
Rank-2	0.975	0.9797	0.4977	0.4699	0.6279	0.4190
Rank-3	0.960	0.9676	0.5210	0.4699	0.6270	0.4230
Ours	0.948 (21)	0.9581 (21)	0.5210 (3)	0.452 (9)	0.6209 (9)	0.4607 (16)

Table 2: Comparison of our results with those on top of the leaderboard. (\*) indicates our rank on the leaderboard in that task.

set for  $n$  epochs. In case of multi task models, all the tasks do not converge at the same rate. Thus, we train multi task models on randomly sampled 8200 texts from `train + public-dev` dataset and validate on the remaining 800 texts. We use early stopping criteria on validation dataset independently for each task.

## 5 Results

We have trained multiple single task and multi task models using basenet LLMs like BERT, DistilBERT, RoBERTa, XLNet, Albert (Lan et al., 2019), Electra (Clark et al., 2020), DeBERTa, and ERNIE-2.0. We also learned ensembles of single task models by either training a classification/regression head on concatenated input embeddings or using weighted aggregate of the models’ predictions. Apart from this, we also explored voting based ensemble of multi-task models. All our models perform comparably on all tasks and the major models are reported in Table 1. We also compare our best model performance with the top 3 submissions on the leaderboard and report it in Table 2.

## 6 Analysis

### 6.1 Data Augmentation

One recurring issue across all our trained models is the high susceptibility to overfitting. Data Augmentation is a widely accepted solution to reduce overfitting by generating slight variants of the given dataset and is extremely useful for a smaller dataset.

One such approach is Masked Language Modelling (MLM), used to perform context-specific data augmentation (Ma, 2019) and has been used in training LLMs. However, following this data augmentation during training has consistently degraded the performance of our models. We hypothesize that this is due to the mismatch be-

tween the contextual meaning and the associated humor/offense. MLM-based augmentation strategies, with models pre-trained to preserve the sentence’s meaning, fail to capture the associated humor/offense.

Often the selection of words in a sentence is responsible for its humor/offensive rating. Replacing such words by their synonyms can change the humor/offense rating substantially. Hence, using such a data augmentation approach during training will inject heavy noise in the ground truth resulting in deteriorated performance.

### 6.2 Correlation across Tasks

Contrary to our belief, we fail to ascertain any direct relationship between the humor controversy and the offense rating prediction task. We compute the mean offense rating for the texts labeled as controversial and for texts marked as non-controversial. The computed mean values are too close to each other to demonstrate any direct correlation conclusively.

### 6.3 Dataset Size

In literature, finetuning LLMs on small size task specific dataset has shown remarkable task performance. However, our single dedicated task models could not perform better than our multi-task model for Task 1b. We attribute this to relatively small size of supervised dataset available for Task 1b in comparison to other tasks. In our multi task models, though we have lesser labeled text for Task 1b, our sentence embeddings are still updated using the complete available dataset. Thus, our multi task model learns underlying distribution better than single task model owing to join learning and shared parameters for task 1b and 2. We believe that this is the main reason for the enhanced performance of our model on Task 1b which has lesser supervised data available in comparison to Task 1a or 2.

## 7 Conclusion

We have presented several experiments using large language models like BERT, XLNet, etc., and their ensembles for humor and offense detection and rating. We also discuss some of the underlying challenges due to the subjective nature of humor and offense detection task. Using these, we explain why standard training practices used to prevent overfitting, like data augmentation, do not work in this context. Our experiments suggest that even though these models can reasonably capture humor and offense, they are still far from understanding every intricacy arising out of subjectivity. To tackle some of the problems highlighted in this paper, a compelling direction would be online data augmentation by alternating between training the embeddings and generating new texts to preserve the humor/offensiveness. Additionally, pretraining these models on datasets annotated by diverse annotators to capture a more comprehensive world knowledge should further help in generalization.

## References

- Issa Annamradnejad. 2020. [ColBERT: Using BERT sentence embedding for humor detection](#).
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Ronan Collobert and Jason Weston. 2008. [A unified architecture for natural language processing: Deep neural networks with multitask learning](#). In *Proceedings of the 25th International Conference on Machine Learning, ICML '08*, page 160–167, New York, NY, USA. Association for Computing Machinery.
- Pierre Colombo, Wojciech Witon, Ashutosh Modi, James Kennedy, and Mubbasir Kapadia. 2019. [Affect-driven dialog generation](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3734–3743, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A Dataset of Fine-Grained Emotions. In *58th Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Martin Docekal, Martin Fajcik, Josef Jon, and Pavel Smrz. 2020. Jokemeter at semeval-2020 task 7: Convolutional humor. *arXiv preprint arXiv:2008.11053*.
- Iuliana Alexandra Fleşcan-Lovin-Arseni, Ramona Andreea Turcu, Cristina Sirbu, Larisa Alexa, Sandra Maria Amarandei, Nichita Herciu, Constantin Scutaru, Diana Trandabăt, and Adrian Iftene. 2017. [#WarTeam at SemEval-2017 task 6: Using neural networks for discovering humorous tweets](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 407–410, Vancouver, Canada. Association for Computational Linguistics.
- Tushar Goswamy, Ishika Singh, Ahsan Barkati, and Ashutosh Modi. 2020. [Adapting a language model for controlled affective text generation](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2787–2801, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. *ArXiv preprint*.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Adilzhan Ismailov. 2019. Humor analysis based on human annotation challenge at iberlef 2019: First-place solution. In *IberLEF@ SEPLN*, pages 160–164.
- Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2019. Sentlr: Linguistic knowledge enhanced language representation for sentiment analysis. *arXiv preprint arXiv:1911.02493*.
- Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A method for stochastic optimization](#).
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019.

- Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
- Xin Li, Lidong Bing, Wenxuan Zhang, and Wai Lam. 2019. Exploiting BERT for end-to-end aspect-based sentiment analysis. *arXiv preprint arXiv:1910.00883*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#).
- Edward Ma. 2019. Nlp augmentation. <https://github.com/makcedward/nlpaug>.
- Lysimachos Maltoudoglou, Andreas Paisios, and Harris Papadopoulos. 2020. [BERT-based conformal predictor for sentiment analysis](#). volume 128 of *Proceedings of Machine Learning Research*, pages 269–284. PMLR.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Aaditya Singh, Shreeshail Hingane, Saim Wani, and Ashutosh Modi. 2021. An end-to-end network for emotion-cause pair extraction. *arXiv preprint arXiv:2103.01544*.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. [Ernie 2.0: A continual pre-training framework for language understanding](#).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).
- Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. ”*Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*”.
- Wojciech Witon, Pierre Colombo, Ashutosh Modi, and Mubbasir Kapadia. 2018. [Disney at IEST 2018: Predicting emotions using an ensemble](#). In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 248–253, Brussels, Belgium. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Huggingface’s transformers: State-of-the-art natural language processing](#).
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.
- Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.
- Yichao Zhou, Jyun-Yu Jiang, Jieyu Zhao, Kai-Wei Chang, and Wei Wang. 2020. ” the boating store had its best sail ever”: Pronunciation-attentive contextualized pun recognition. *arXiv preprint arXiv:2004.14457*.

# RoMa at SemEval-2021 Task 7: A Transformer-based Approach for Detecting and Rating Humor and Offense

**Roberto Labadie Tamayo**  
Universidad de Oriente  
Cuba  
rlabadiet@gmail.com

**Mariano J. Rodriguez Cisneros**  
Universidad de Oriente  
Cuba  
mjasoncuba@gmail.com

**Reynier Ortega-Bueno**  
PRHLT Research Center  
Universitat Politècnica de València  
Valencia Spain  
rortega@prhlt.upv.es

**Paolo Rosso**  
PRHLT Research Center  
Universitat Politècnica de València  
Valencia Spain  
prossod@dsic.upv.es

## Abstract

In this paper we describe the systems used by the RoMa team in the shared task on *Detecting and Rating Humor and Offense (HaHackathon) at SemEval 2021*. Our systems rely on data representations learned through fine-tuned neural language models. Particularly, we explore two distinct architectures. The first one is based on a Siamese Neural Network (SNN) combined with a graph-based clustering method. The SNN model is used for learning a latent space where instances of humor and non-humor can be distinguished. The clustering method is applied to build prototypes of both classes which are used for training and classifying new messages. The second one combines neural language model representations with a linear regression model which makes the final ratings. Our systems achieved the best results for humor classification using model one, whereas for offensive and humor rating the second model obtained better performance. In the case of the controversial humor prediction, the most significant improvement was achieved by a fine-tuning of the neural language model. In general, the results achieved are encouraging and give us a starting point for further improvements.

## 1 Introduction

Detecting humor has become a popular research field at the same time that the bad phenomenon of offensiveness spreading exaggeratedly grows in social media. In this scenario it is very frequent to find out alarming volumes of heterogeneous data such as textual messages, images, advertisements, etc., that harm some age groups, ethnicity, sexual gender or other demographic characteristics (Betul

Keles and Niall McCrae and Annmarie Grealish, 2020). Most of these harmful contents are often masquerade as innocent jokes or simply as a funny content. Therefore, it is crucial to shed light on the commonalities and differences between both phenomena in order to properly addressing the challenge of computationally distinguishing humorous messages from aggressive or offensive ones. Recognizing humorous and offensive utterances on written messages is a very difficult task for human beings and even more for computers (Waseem, 2016). These difficulties increase when the textual messages are isolated from the context in which they are produced. Additional knowledge from gestures, prosody features, visual content, situational environment and sociocultural rules play an important role in how humans properly understand the real meaning behind funny and hateful contents. All this makes humor recognition and offensiveness detection challenging tasks within Natural Language Processing (NLP) and Human-Computer Interaction (HCI). On this line, the *Task 7, HaHackathon: Detecting and Rating Humor and Offense at SemEval-2021* aims at computationally recognizing humor and offensiveness in English tweets (Meaney et al., 2021).

To address the four subtasks launched in *HaHackathon* we propose two distinct architectures which rely on neural language model based representation (*deep-representation*), particularly learned by Transformer architectures. Our first architecture combines the learned representation with a SNN in order to learn in automatically way a metric for discriminating a pair of messages of

the same class from a pair of messages of different classes. Also, we considered applying a graph-based clustering method to each class independently for creating representative prototypes. These prototypes were used to build the training and testing pairs. Our second architecture relied on the principle of fusing representations. For that, the *deep-representation* are mixed with linguistic information (*linguistic-representation*) and given as inputs to a linear regression model which is specialized in predicting humor and offensive scores. The paper is organized as follows: in Section 2 we briefly introduce the description of the four subtasks. Section 3 presents our proposed architectures and gives details about their modules. In Section 4 are described the experiments and results. Finally, we present our conclusions and provide interesting directions that we plan to explore in future work. The source code associated with this paper is online available on GitHub: [https://github.com/mjason98/semeval21\\_humor](https://github.com/mjason98/semeval21_humor).

## 2 Task Descriptions

We investigated the performance of our proposed architectures in the four subtasks introduced in *Ha-Hackathon*: i) given a tweet determining whether it is humorous or not (*subtask 1a*); ii) given a tweet predicting the humor rate in the range of 0 to 5, where 0 indicates that it is not a funny message and 5 indicates that the message is strong humorous (*subtask 1b*); iii) given a tweet determining whether it is considered as controversial (i.e., it is rated with highly variable values of humor from one annotator to another) (*subtask 1c*); the last subtask, iv) given a tweet predicting its offensiveness rating in a range of 0 to 5, where 0 indicates the tweet does not contain any kind of offensiveness and 5 indicates that the message is strong offensive (*subtask 2*).

Organizers provided a dataset for training and test labeled according to the objectives of each subtask. The whole dataset was manually annotated by several annotators in order to minimize the noise in the data and increase the agreement in the annotation procedure. The dataset is composed by 8000 tweets for training and 1000 tweets for testing purposes, respectively. The training set contains 3068 funny and 4932 non-funny tweets. This slight imbalance in the training set imposes an additional difficulty to the learning algorithm for accurately predicting the funny messages. The

problem increases in the task of controversial humor prediction where only 2465 tweets are labeled as controversial and the remainder 5535 are non-controversial. The most complex scenario regarding the data distribution is appreciated in the tasks of humor and offensiveness rating. At a first glance on the Figure 1 can be inferred that the majority of the offensive scores are accumulated in the interval (0,1). As a consequence of that, the tweets which

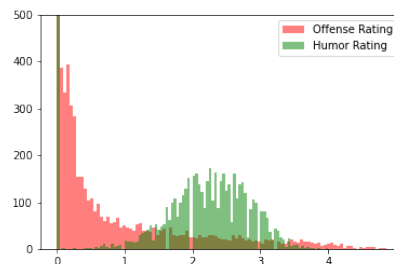


Figure 1: Histograms of humor and offensiveness score distribution

are not offensive at all or those with scores closer to zero are over-represented whereas the tweets with strong offensive content are under-represented in the dataset. Therefore, from the learning perspective, it is more difficult to score tweets which strong offensive content. Conversely to this scenario, the funniness scores are distributed more uniform. Also, it is important to highlight that tweets with offensive scores greater than 0 in most cases also were scored as funny tweets. This relation reveals the usage of some humor devices as a way for masqueraded offensive messages.

## 3 Our Proposals

In this section we present the proposed models and provide details about their modules. Our models have a modular structure. They are composed of both, an encoder module (*Encoder*) and a prediction module (*Classifier*), which are trained independently. Particularly, we evaluate two distinct methods for the classification module. The first one is based on a Siamese neural network and the second one relies on fusing representations and training a linear regression model.

### 3.1 Encoder Modules

The Encoder plays an important role because it is concerned with learning an abstract representation that vanishes the colinearity between its features

and compresses the textual information on a single dense vector. In our proposal, the encoders are based on Transformer models (TM), specifically on RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2020) architectures. Moreover, we employed BERTweet (Nguyen et al., 2020) which is based on the structure and pre-training procedure like RoBERTa, but using an English tweets corpus that makes it easier to fine-tune on NLP tasks where the texts are short and informal.

For fine-tuning the TM-based encoders we add up an intermediate layer that receives the vectors from the output sequence of the TM. On this sequence of vectors, we explore three variants for selecting the best way of representing the message: i) the vector in the first position (associated to the *CLS* token), ii) the normalized sum of all vector in the sequence, and iii) the vector in the last position of the sequence. On this layer, we stacked an output layer that makes the final prediction for the targeted task. For that purpose, we follow the strategy proposed in the Universal Language Model Fine-Tuning (ULMFiT) (Howard and Ruder, 2018). For each layer of the TM a different learning rate is set up, increasing it using a multiplier while the neural network gets deeper. This multiplier increases 0.1 points from a layer  $L_i$  to another  $L_{i+1}$ . We use this dynamic learning rate to keep most information from the pre-training at shallow layers and biasing the deeper ones to learn about the specific tasks.

On the humor predicting subtask the BERTweet encoder was employed, whereas for offensiveness rating the three TMs were considered and trained using a multitask learning strategy for predicting offensive scores and irony together. Particularly, for irony detection we used the data proposed in *Task 3: Irony Detection Task at SemEval 2018* (Van Hee et al., 2018).

## 3.2 Classification Modules

In this section we describe the architecture of the two proposed classification modules.

### 3.2.1 SiaNet

To address the humor detection task we propose a SNN (Koch et al., 2015; Bromley et al., 1993) whose functionality lies on extracting features from the input messages, in such a way that a pair of messages belonging to the same class are closer and in case of belonging to opposite class move away w.r.t a distance function. In this work we use the Euclidean distance. The distance learned by this

network is used as a criterion to determine, given an unlabeled message, whether it is more likely to belong to the positive class (e.g. humorous) than to the negative class (e.g. non-humorous). For that purpose, we define in each class a set of prototypes which are used to compare against the unlabeled message. These prototypes are obtained by means of a graph-based clustering method. After having the clusters, for each of them is selected a prototype (real message), which is able to represent the most information contained on that group.

### Prototype Selection Strategy

The SiaNet model requires a pair of messages as input in both training and test phases. During the training stage, pairs of two labeled messages are used, and in the test phase, the label of a new message is predicted considering its similarity with positives (humorous) and negatives (non-humorous) messages. As consequence, the methods employed to obtain the pairs and select the humor and non-humor messages for comparing at the training stage, impact directly on the learning process of the model.

In this work, instead of sampling positive and negative messages randomly, we propose to include an additional step that aims at obtaining prototypical instances (*prototypes*, henceforth) of both classes. For that, we build a graph of  $\beta$ -distance, analogous to the  $\beta$ -similarity graphs proposed in (Garcia, 2005), for the humor ( $G_P$ ) and non-humor ( $G_N$ ) classes. The nodes in the graphs represent the tweets from the training set and the edges joining two nodes are weighted with the distance between them.

In the  $\beta$ -distance's graphs the edges with weights greater than the threshold  $\beta$  are removed, allowing only the closest representations being in the same connected subgraph. Notice that, the representation of the messages associated to nodes are obtained from the Encoder module. Afterwards, we detect communities on the  $\beta$ -distance's graph  $G_P$  and  $G_N$  respectively, using the InfoMap (Edler et al., 2020) algorithm based on the map equation (Rosvall et al., 2009). The map equation is a flow-based and information-theoretic method. By minimizing it over all the possible network partitions, InfoMap reveals important aspects of the network structure with respect to the dynamics on the network.

As result it is obtained a set of subgraphs  $g_P^i \in G_P$  and  $g_N^i \in G_N$  and the nodes they contain with their respective flow values. For each subgraph

$g_C^i$ ,  $C \in [P, N]$  we select the node  $x_{max}$  with the highest flow value. We assume that this node acts as a representative node for  $g_C^i$ , and consequently it is also a prototypical message for the class  $C$ . All prototypical messages for the humorous and non-humorous classes are obtained and defined as Humor Prototype Set ( $P_{Set}$ ) and Non-Humor Prototype Set ( $N_{Set}$ ) respectively. In Figure 2 are depicted the projection of each class messages, and their respective prototypes.

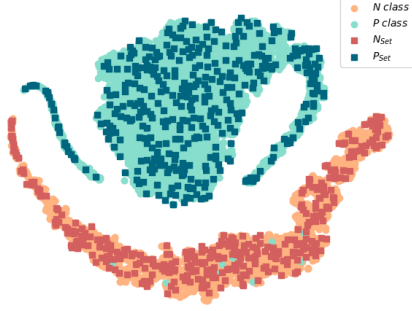


Figure 2: Scatter encoder representations per class with the identified prototypes

### Siamese Neural Net Architecture

The network architecture consists of two input messages and one output that indicates how distant they are according to their representation (Bromley et al., 1993). Both messages are encoded by using the fine-tuned Transformers model (see Section. 3.1). Later, each input is passed through two dense layers (with 64 hidden neurons), which map the encoding to a smaller dimension by learning specific features. We must annotate that both input messages are fed to the same two dense layers (i.e., the new encodings are computed using the same weights in both cases). Later, the representations of the messages are compared to each other through a distance metric. The specific features the model learns to extract, make that messages representations corresponding to opposite classes have a distance greater than the threshold defined in the loss function used. Particularly, we used the *Contrastive Loss* (Hadsell et al., 2006) with a threshold of 0.85, this value was set empirically.

For training the SNN, the dataset needs to be processed for constructing pairs of messages from the same class and pairs of messages from distinct classes. Once defined the sets of prototypes  $P_{Set}$  and  $N_{Set}$  (as described in the Prototype Selection Strategy), we create training examples associated

to each message  $x$  into the training dataset, with  $x \notin P_{Set} \cup N_{Set}$ . For that, we sampled randomly  $k$  *intra-class* examples, by pairing  $x$  with prototypes from its class, and generate  $m$  *inter-class* examples pairing  $x$  with their closest prototypes from the contrary class.

During the test phase, given an unlabeled message, we obtain the encoding of  $z$  by using the Encoder module. After that, we predict the distance of  $z$  with respect to the prototypes in the  $P_{Set}$  and  $N_{Set}$  using the SNN. Based on the previously computed distances, we evaluate two rules for deciding whether  $z$  should be classified under the humorous or non-humorous classes:

i) *Minimum*, we assign  $z$  to the class of its nearest prototype as follows:

$$\hat{y} = \arg \min_i \{SNN(z, x_{i,j})\} \quad (1)$$

where  $x_{i,j}$  is the prototype message  $j$  with label  $i = \{0, 1\}$ .

ii) *Mean*, we assign  $z$  to the class of the Prototype Set with lowest average distance:

$$S_i = \frac{1}{C_i} \sum_{j=1}^{C_i} SNN(z, x_{i,j})$$

$$\hat{y} = \arg \min_i \{S_i\} \quad (2)$$

where  $C_i \in \{|P_{Set}|, |N_{Set}|\}$  and  $x_{i,j}$  is the prototype message  $j$  with label  $i = \{0, 1\}$ .

### 3.2.2 Multiview-based Linear Regression Module

Ensemble methods usually combine data representations or the decisions of multiple models to obtain improved results over those obtained individually. These decisions are made from valuable features extracted by models' intermediate layers, which vary depending on their architecture and the dataset they have been trained on.

Combining all those information into a single prediction unit instead of synthesized predictions, is consistent if we seek to take into account different views of the information, especially when dealing with such complex and subjective tasks as offensiveness detection and in general sentiment analysis are.

We propose to fusing four distinct representations of the tweets and use this mixing deep-features for training a linear regression method. Three of the representations are based on fine-tuned transformer encoders and the other is based on affective features.



## Encoder Settings

Considering the underlying relation between humorous and offensive language observed in the HaHackathon dataset (please, see Figure 1), the relation among offensiveness with other forms of toxic speech (e.g. aggressiveness and hate) presented in (Poletto et al., 2020) and the common usage of figurative devices like irony in social media for communicating indirectly hateful messages (Cignarella et al., 2018; Frenda, 2018). We fine-tuned the RoBERTa-based models on the dataset provided in the shared tasks HatEval 2019 (Basile et al., 2019), OffensEval 2019 (Zampieri et al., 2019), Irony Detection Task at SemEval 2018 (Van Hee et al., 2018) and HaHackathon itself. The fine-tuning was carried out using a smooth learning rate on the Masked Language Modeling (*MLM*) task. We masked randomly 15% of the tokens from each message, and fit them for three epochs, following the strategy proposed in (Liu et al., 2019).

For training the Encoders to address the HaHackathon specific target, the placed intermediate layer after the encoder heads, is fed with the concatenation of the three variants to get the TM output (see Section. 3.1). This layer is the one employed to obtain the message encodings. We combine the offensiveness rating in HaHackathon with the labels of irony in the dataset of *SemEval 2018 Task 3* (Van Hee et al., 2018). This idea relies on the observed relation between humor and offensiveness ratings within the provided data, where many offensive messages can be considered as ironic or harmful forms of humor (see last two examples in Table 4).

To avoid outliers in the dataset for misleading the training process, we employed the *Minkowski error* (Bishop, 1995) in the regression subtask, which is less sensitive to outliers than the standard mean squared error. It is defined as follows:

$$Error_{Minkowski} = \frac{\sum(|y - \hat{y}|)^{kc}}{n} \quad (3)$$

Where  $y$  is the label for one example,  $\hat{y}$  is the predicted value,  $n$  the number of examples and  $kc$  the Minkowski coefficient which we set to 1.4 empirically.

## The Affective Features

Conversely to the three representations above, the Affective Features representation was obtained from a word-level recurrent neural network, trying to capture how affective information from different

dimensions flows along the messages (Kar et al., 2018). For this purpose, we constructed an embedding matrix whose features were based on an affective information set proposed by (Farías et al., 2016) containing basic emotions (i.e., sadness, surprise, fear, etc). The embedding vectors involved 52 components between binary and no binary values, and the vocabulary was built from the affective resources, hence words not expressing emotional charge at all were encoded with the null vector.

The information obtained from this embedding for a message was fed into an BiLSTM (*Bidirectional Long Short Term Memory*) architecture similar to ELMO (Peters et al., 2018). Deeper BiLSTM output was fed, to an intermediate layer to condense the information passed later to the output layers. For training this model we used a multitask approach focusing on predicting how offensive a message is, as well as how funny it is, by using the data provided for HaHackathon.

## Linear Regression

As we can be observed in Figure 3, the encoding provided by transformer models differ regarding the space region in which the offensive features are projected. We can infer that one representation helps the others by providing information not captured simultaneously.

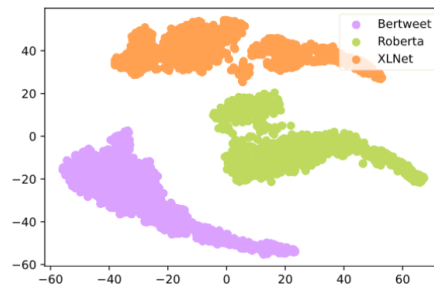


Figure 3: Scatter encoders representations

Considering that, there is no co-linearity between the features extracted from one encoder to another. We hypothesize they can be combined through a parsimonious model to prevent overfitting. Based on that, we decided to employ a Ridge Regression Model, setting the  $\alpha$  hyper-parameter employed for the L2 regularization on the loss function to 1.0. During the experiments we also construct another ensemble based on Recurrent Neural Networks (RNN) which receive all four encodings and treat them as a sequence of the message. The elements of this sequence are weighted through an Addi-

tive Attention layer and combined employing an LSTM layer in order to decide from a time step *i.e.*, from one encoding to another, which of the features must be kept through the entire sequence analysis *i.e.*, all four encodings. The output of this layer is then fed into two parallel output layers to predict whether a message is offensive or not and its offensiveness degree.

The data over-representation for messages with offensiveness rating equal to 0 makes that, from the standpoint of deciding whether a message is offensive or not, the data be balanced (*i.e.*, labeling the messages with offensiveness rating higher than 0 as offensive). This allowed to us using the alternative classification task in the multitask learning approach employed for this model, taking into account that it helps to learn common features among these offensiveness-related tasks.

## 4 Experiments and Results

In this section we describe the conducted experiments for evaluating the performance of our systems on HaHackathon development dataset (*dev-dataset*). For that, we employed the metrics proposed by the task organizers, *F1-score* over the positive class and accuracy (*Acc*) for classification subtasks, and the Root Mean Squared Error (*RMSE*) for regression subtasks.

### Encoder Modules

The SiaNet model and the Ridge Regression model are fed with information of the messages extracted through their respective encoder modules, this makes our first effort focused on tuning them for obtaining the best representation. For both approaches, SiaNet and Ridge Regressor, the encoders were optimized using the RMSprop method (Hinton et al., 2012).

Firstly, for obtaining the multi-viewed representation of the messages, the RoBERTa, BERTweet and XLNet encoders were fine-tuned using (*MLM*) unsupervised learning. For that, we considered three additional related-datasets (Basile et al., 2019; Zampieri et al., 2019; Van Hee et al., 2018) and the HaHackathon dataset itself. Afterwards, since the multi-viewed representation was constructed for rating offensiveness, the three models were trained specifically for this regression task by exploring two main ideas based on multitask learning strategy (*MTL*): i) The first one aims at capturing the

information shared among the four subtasks proposed in the HaHackathon dataset; ii) The second one, aims at capturing the indirect negative speech behind humorous messages, for that we introduced the irony prediction task (*Irony*) combined with offensiveness prediction. Specifically, we used the dataset proposed in (Van Hee et al., 2018) for addressing the irony prediction task.

Table 1 shows the results of applying both strategies for each transformer encoder. As can be observed, by making the model to extract features also useful for irony detection we achieved the best performance. Nevertheless, the first strategy

Model	Strategy		
	<i>HAHA</i>	<i>Irony</i>	<i>No MTL</i>
BERTweet	0.70	0.65	0.81
RoBERTa	0.75	0.63	0.67
XLNet	0.69	0.68	0.70

Table 1: MTL strategies for offensiveness rating subtask. *HAHA* refers to MTL with all HaHackathon subtasks and *Irony* refers to MTL with irony detection task

yields our best result at predicting whether or not a message can be considered as controversially humorous. We also tried to avoid using MTL with the three transformers encoders, fine-tuning them for the offensiveness regression subtask, but in terms of RMSE the performance decreased on 0.07 in average.

Similarly it happened when it was not accomplished the *MLM* fine-tuning. The error was slightly increased for RoBERTa from 0.58 to 0.64 when this stage was avoided and for BERTweet, it increased from 0.65 to 0.91. We hypothesize this technique helped the model to reduce the impact of isolated offensive terms, which may influence the regression stage on messages that are not even offensive.

For fine-tuning the encoder module of SiaNet we explored if it was more convenient to set a single learning rate for the whole model or follow the ULMFiT strategy addressing the humor prediction task. The second approach obtained the best performance in terms of F1-Score/Acc with (0.94/0.92) w.r.t (0.90/0.88) reached by the first one. We also tried to apply MTL to this approach, but this did not yield any improvement, reaching 0.93/0.91.

## Prediction Modules

For classifying unlabeled tweets with SiaNet we evaluated the two methods described in Section 3.2.1 alongside the upper bound of clusters extracted from  $G_P$  and  $G_N$  by InfoMap (50, 250 and 300 clusters) and how the TM output sequence was used according to the strategies described in Section 3.1. Among the combinations resulting from that evaluation, the best-performed was the one involving the *minimum* criterion for labeling the messages, the highest upper bound for allowed instances on  $P_{Set}$  and  $N_{Set}$  respectively (300) and taking the normalized sum of the TM output sequence, reaching under F-Score/Acc the measurements (0.9505/0.9370).

Also, we added Gaussian noise to the encoding inputs for decreasing overfitting when training the Siamese as part of the conducted experiments, resulting on improving the loss in the dev-set from 0.11 when the noise is not added to 0.06.

In the training phase of the Linear Ridge regression method we evaluated the impact of the distinct representations on the performance of our model. Looking at Table 2, we noted that each transformer

model	XLNet	RoB	BT	AF	Off
	+	+	+	+	<b>0.55</b>
	+	+	+	-	0.61
Ridge	-	+	+	-	0.65
	+	+	-	+	0.58
	+	-	+	+	0.59
	-	+	+	+	0.62
LSTM	+	+	+	+	<b>0.55</b>
LSTM-Att	+	+	+	+	0.57

Table 2: Feature representation combination through the ensemble

encoder played an important role in characterizing the messages, also the affective features captured important information about the offensive language, which helped in each combination. The LSTM based models also had a good performance when combining all the representations, especially the one with no attention mechanism.

Summarizing, participating in HaHackathon we addressed the humor prediction task with the SiaNet model. For the humor rating subtask we used the Multiview-based Linear Ridge Regression model, fine-tuning the transformer encoders under the humor and offensiveness rating subtasks simultaneously after applying *MLM*. The controversy

humor prediction subtask was addressed through the BERTweet model using *MTL* with all four sub-tasks from HaHackathon. Finally, the offensiveness rating was predicted by the Multiview-based Linear Ridge Regression, but fine-tuning the encoders with *MTL* and combining offensiveness rating sub-task with irony detection.

## 4.1 Error Analysis

In the humor prediction subtask, we found out that more than 40% of prototypes obtained from the humorous class have the structure *question?argumentation* (Q?A, see Table 3). We hypothesize that some tweets were misclassified as humorous due to sharing this structure with positive prototypes. In fact, within the examples labeled by our architecture as funny when they were not, the ones having this structure represented the 38% of this type of misclassification.

Tweet
What do you call an Asian guy that always shows up before he needs to? Earl Lee
Why did the slave go to college? So he could pickup his Master’s degree.
What do you call a 60-year old whose puberty just started? A late boomer.

Table 3: Prototype tweets annotated as humor with the structure of Q?A

For the offensiveness prediction task, the most critical failures (i.e., absolute difference between the real value and the predicted one) were analyzed from two standpoints: first when the model predicts a lower value than the real one as the first two examples in Table 4 or a higher value as in the last two cases. As we can observed how it happened in the most mispredicted examples, the model gives higher offensiveness values to messages containing phrases that characterize social groups usually being a target of hate spreading or bullying on social media. This is possibly caused by the origin of data used for pre-training the transformer encoders, which were in charge of finding an encoding for the tweets.

## 4.2 Official Results

Regarding the official results on the test set, we made submissions in all four sub-tasks. The baseline proposed by the organizers consisted of a Naive Bayes model with bag of words features

Tweet	Value	Predicted
What do you call a <b>homosexual man</b> on a <b>wheel chair</b> ? A human being	0.15	2.5
What do you call it when two female spies fall in love? <b>Lesbianage</b>	0.6	1.89
Wanna hear a <b>joke</b> ? <b>Women’s rights</b> .	3.35	1.79
What belongs to me but is used the most by others? <b>My ex-wife</b>	1.9	0.43

Table 4: Some examples mispredicted by our model

and Support Vector Regression for the classification and regression subtasks respectively.

In subtask 1a we ranked at place 22<sup>nd</sup> among 58 teams, with F-Score/Acc of 0.948/0.9576, whereas the best system reached 0.982/0.9854. In subtask 1b with a RMSE of 0.5905 and among 50 teams we ranked at 30<sup>th</sup> place and the best system had an RMSE of 0.4959. For subtask 1c we obtained the 10<sup>th</sup> position from 36 teams, our F-Score/Acc was 0.4732/0.6197 and the best system obtained 0.4943/0.6302. Finally, in subtask 2 we were the 14<sup>th</sup> team of 48 in total, with a RMSE of 0.4532 with a difference from the best ranked system of 0.0412.

## 5 Conclusions and Future Work

In this work we presented two models for addressing humor and offensiveness prediction in English tweets. Both models employ the deep-representations learned by Transformers methods for encoding the messages. The first model is based on a Siamese Neural Network combined with a graph-based clustering method. The second model combines feature representations learned by three transformers language models with affective features captured by an BiLSTM-based model. These representations are used to train a linear regression model. The achieved results show that the Siamese architecture outperformed the fine-tuned Transformer models for humor detection task. The performance of this architecture relies on how the tweets are represented by the encoder and the strategy to find the Positive and Negative sets of prototypes. In the second model, the affective features play an important role to determine the offensiveness scores with any combination of features learned by the state-of-the-art language models, showing that they successfully captured underlying affective cues present in offensive and funny speech. We plan to investigate two interesting directions as future works. The first direction is an in-depth study of the harmfulness of humor on human stereotypes taking advantage of the over-

lapping between offensiveness and humor in the HaHackathon dataset. The second one is an exhaustive analysis of clustering methods for building prototypes and how they may influence the learning of the Siamese Neural Network for humor prediction.

## Acknowledgements

The work of the last two authors was in the framework of the research project MIS-MIS-FAKENHATE on MISinformation and MIScommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31), funded by Spanish Ministry of Science and Innovation, and DeepPattern (PROMETEO/2019/121), funded by the Generalitat Valenciana.

## References

- Betul Keles and Niall McCrae and Annmarie Grealish . 2020. *A systematic review: the influence of social media on depression, anxiety and psychological distress in adolescents*. *International Journal of Adolescence and Youth*, 25(1):79–93.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Nozza Debora, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, Manuela Sanguinetti, et al. 2019. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceeding of the 13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.
- Christopher M. Bishop. 1995. *Neural Networks for Pattern Recognition*. page 208. Oxford University Press, Inc., USA.
- Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a “siamese” time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688.
- Alessandra Cignarella Cignarella, Simona Frenda, Valerio Basile, Cristina Bosco, Viviana Patti, and Paolo Rosso. 2018. Overview of the Evalita 2018 Task on Irony Detection in Italian Tweets (IronITA). In *Proceedings of the 6th evaluation campaign of Natural*

- Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.
- Daniel Edler, Eriksson Anton, and Martin Rosvall. 2020. The MapEquation software package. URL: <https://mapequation.org>.
- Delia Irazú Hernández Farías, Viviana Patti, and Paolo Rosso. 2016. Irony detection in twitter: The role of affective content. *ACM Transactions on Internet Technology (TOIT)*, 16(3):1–24.
- Simona Frenda. 2018. The role of sarcasm in hate speech. A multilingual perspective. In *Doctoral Symposium of the XXXIV International Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, pages 13–17. Lloret, E.; Saquete, E.; Martínez-Barco, P.; Moreno, I.
- Reynaldo Jose Gil Garcia. 2005. *Algoritmos de agrupamiento sobre grafos y su paralelización*. Ph.D. thesis, Universidad Jaume I.
- Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.
- Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Lecture 6a overview of mini-batch gradient descent. *Coursera Lecture slides* <https://class.coursera.org/neuralnets-2012-001/lecture>, [Online].
- Jeremy Howard and Sebastian Ruder. 2018. **Universal Language Model Fine-tuning for Text Classification**. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.
- Sudipta Kar, Suraj Maharjan, and Tamar Solorio. 2018. **Folksonomication: Predicting Tags for Movies from Plot Synopses using Emotion Flow Encoded Neural Network**. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2879–2891, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. *CoRR*, abs/1907.11692.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 Task 7, HaHackathon, Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. **BERTweet: A pre-trained language model for English Tweets**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. **Deep Contextualized Word Representations**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Fabio Poletto, Valerio Basile, Manuela Sanguinetti, Cristina Bosco, and Viviana Patti. 2020. **Resources and benchmark corpora for hate speech detection: a systematic review**. In *Language Resources and Evaluation*, pages 1–47. Springer Netherlands.
- Martin Rosvall, Daniel Axelsson, and Carl T Bergstrom. 2009. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23.
- Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. **SemEval-2018 Task 3: Irony Detection in English Tweets**. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.
- Zeeraq Waseem. 2016. **Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter**. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. **XLNet: Generalized Autoregressive Pretraining for Language Understanding**. *arXiv preprint arXiv:1906.08237*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. **SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval)**. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

# SemEval-2021 Task 8: MeasEval – Extracting Counts and Measurements and their Related Contexts

Corey A Harper<sup>1,2</sup>, Jessica Cox<sup>1</sup>, Curt Kohler<sup>1</sup>, Antony Scerri<sup>1</sup>,  
Ron Daniel, Jr.<sup>1</sup> and Paul Groth<sup>2</sup>

<sup>1</sup>Elsevier Labs, Suite 800, 230 Park Avenue, New York, NY 10169, USA

<sup>1</sup>{*c.harper, j.cox, c.kohler, a.scerri, r.daniel*}@elsevier.com

<sup>2</sup>University of Amsterdam, Postbus 94323 / 1090 GH, Amsterdam

<sup>2</sup>{*c.a.harper, p.t.groth*}@uva.nl

## Abstract

We describe MeasEval, a SemEval task of extracting counts, measurements, and related context from scientific documents, which is of significant importance to the creation of Knowledge Graphs that distill information from the scientific literature. This is a new task in 2021, for which over 75 submissions from 25 participants were received. We expect the data developed for this task and the findings reported to be valuable to the scientific knowledge extraction, metrology, and automated knowledge base construction communities.

## 1 Introduction

Counts and measurements are an important part of scientific discourse (Rijgersberg et al., 2011). It is relatively easy to find measurements in text (Foppiano et al., 2019a), but a bare measurement like 17mg is not informative without knowing what it is referring to. For example, it is important to know whether a quantity is 17mg of a medicine dosage or 17mg of concrete additive. Only recently have attempts been made to identify the named entity and property being measured (Hundman and Maamann, 2017). Extracting such information is challenging because the way scientists write can be ambiguous and inconsistent. Furthermore, the location of this information relative to the measurement can vary greatly, and might even be in a different sentence.

Being able to extract measurement information automatically can enable the construction of databases of measured properties. Such databases are important in biomedicine (Hao et al., 2016), engineering (Foppiano et al., 2019a), and other scientific disciplines (Bergmann et al., 2017), but the approaches used for populating these databases do not generalize widely. Furthermore, knowledge graphs (Hogan et al., 2021) frequently aggregate quantitative data reported in the literature and are often

built through a largely manual curation process. Examples include: LITTERBASE<sup>1</sup> (Bergmann et al., 2017), which aggregates observations of marine litter distribution; NeuroElectro<sup>2</sup> (Tripathy et al., 2014), which collects information on electrophysiological properties of neurons; and various model organism databases like the Zebrafish Information Network<sup>3</sup> (Sprague, 2006), which provide summaries of gene information.

Beyond knowledge graphs and curated databases, clinical health contexts often require extraction of measured values for lab results and patient observations. Moreover, scientific research frequently relies on precise measurements for reproducibility of experimental methods (Kaiser, 2018). Measured property extraction could be of value in many other contexts, such as fact checking and news validation or in statistical analysis for public policy (Einav and Levin, 2014).

Research in information extraction and knowledge graph creation has concentrated on forming triples by extracting entities and relations (Konstantinova, 2014). Little attention has been paid to the extraction of measured properties, entities, and conditions or contexts, yet these elements are needed to place measurements into a database and for their subsequent use in comparison and calculation. Units and measures are an important part of the semantic web, though research has largely been focused on ontology design (Rijgersberg et al., 2013). There is, thus, a need for understanding the state of the art on this important task.

The aim of this paper is to introduce the MeasEval shared task for the extraction of counts, measurements, and related context from English-language scientific documents, as well as to present an analysis of the results of participant systems on

<sup>1</sup><https://litterbase.awi.de/>

<sup>2</sup><https://neuroelectro.org/>

<sup>3</sup><http://zfin.org/>

the task.

The rest of this paper is organized as follows. We begin with a description of related work. This is followed by the description of the task itself (Section 3) and the associated data and annotation procedure (Section 4). The evaluation regime is detailed in Section 5 including baselines. Subsequently, we present an analysis of the results of the systems on the task. Finally, we summarize the various participating systems approaches and conclude.

## 2 Related Work

There is a substantial body of work discussing units of measurement, ontologies to describe them, systems designed to extract them, as well as related work on knowledge graphs of numerical attributes. Automated extraction of measured quantities, such as 520 +/- 8 items/kg, is straightforward and many tools exist to perform this task (Foppiano et al., 2019b; Deus et al., 2017; Hao et al., 2016). To build a knowledge graph, we must put these measurements in context. We need to determine the properties being measured (e.g. abundances), the entities that exhibit those properties (e.g. the Maowei Sea), and possible qualifying conditions under which measurements are obtained (e.g. the date and depth of the sampling). These properties, entities, and conditions can then be mapped to those that are used in the knowledge graph, so that the measurements can be normalized into a common system.

There are a number of ontologies that cover units of measurement, such as Quantities, Units, Dimensions, and Types Ontologies (QUDT)<sup>4</sup> and the Ontology of Units of Measure and Related Concepts (OM) (Rijgersberg et al., 2013). These and others are discussed in a survey paper by (Steinberg et al., 2017). Most of these ontologies focus on conversion between different systems of measurement, and on classifying types of measurement or domain of application, but do not necessarily address the “thing” being measured. The Joint Committee for Guides in Metrology’s (JCGM) International Vocabulary of Metrology covers this in slightly more depth, discussing measurement units and quantity values, then talking about quantities themselves, which it defines as a “property of a phenomenon, body, or substance” (Joint Committee for Guides in Metrology, 2012). We find that this nomenclature, while precise, is likely to be con-

fusing to non-metrologists from both an evaluation and annotation perspective, so to support the data annotation process for this task we use a simplified nomenclature.

Metrology research in the Semantic Web community is often focused on ontology alignment for Units of Measurement ontologies. Kaladevi et al. (2016) look at aligning unit ontologies to support merging data across many weather information systems, while Do and Pauwels (2013) more generally look at using MathML for aligning unit ontologies. Efforts around designing linked data models for semantic sensor streams for the Internet of Things also utilize the Units of Measurement ontology for representing measurement information (Barnaghi et al., 2013). None of this work addresses extraction of measurements and their contexts nor building knowledge graphs from such information.

Other research explores creating databases of numeric attributes. Kotnis and Garcia-Duran (2019) infer new values using linear regression for neighboring entities in a knowledge graph. Gupta et al. (2015) use a logistic regression with distributional vectors. Davidov and Rappoport (2010) use a system of averages and boundary values to infer an estimated numeric attribute value. Rather than imputing new values from related entities, MeasEval starts from a value and puts it into the context of measured entities and measured properties, working toward a knowledge representation of numeric data.

## 3 Task Description

MeasEval is an entity recognition and semantic relation extraction task focused on finding counts and measurements, attributes of those quantities, and additional information including measured entities, properties, and measurement contexts.

MeasEval is composed of five sub-tasks that cover span extraction, classification, and relation extraction, including cross-sentence relations. Given a paragraph from a scientific text:

- For each paragraph of text, identify all spans containing quantities (e.g. 12 kg). Quantities are treated as strings, and are not converted or normalized.
- For each identified Quantity, identify the Unit of Measurement (e.g. kg), if one exists. For each Quantity classify additional value Modifiers (e.g. count, range, approximate, mean,

<sup>4</sup><http://www.qudt.org/>

etc.) that apply to the Quantity.

- For each identified Quantity, identify the Measured Entity (e.g. bed inventory) it applies to (if one exists) and mark its span. If an associated Measured Property (e.g. concentration) also exists, identify it and mark its span.
- Identify and mark the span of any Qualifier (e.g. after incubation) that is needed to record additional related context to either validate or understand each identified Quantity.
- Identify relationships between Quantity, Measured Entity, Measured Property, and Qualifier spans using the HasQuantity, HasProperty, and Qualifies relation types.

More detailed definitions can be found by reviewing the MeasEval Annotation Guidelines.<sup>5</sup> We describe each of the elements to be extracted in more detail in the next section.

## 4 Annotated Data

### 4.1 Data Model

As shown in Figure 1, the MeasEval annotation model consists of Quantities, MeasuredEntities, MeasuredProperties, and Qualifiers. A Quantity can be either a count or a measurement, with measurements being composed of a Unit and a Value. Values also can have additional attributes such as “isMean”, “isApproximate”, or “isRange”. Quantities can be directly related to a MeasuredEntity, or can be indirectly related to a MeasuredEntity via a MeasuredProperty. Qualifiers provide additional information that is required to interpret the measurement. These include things like the pressure at which a boiling point was observed, or the depth and location where an ocean sample was taken. Since texts may contain different parts of this information, all relationships are optional. A MeasuredEntity can be related to a MeasuredProperty or a Quantity, a MeasuredProperty can be related a Quantity, and a Qualifier can have a relationship to any span.

### 4.2 Corpus and Annotations

Annotations are drawn from 110 CC-BY licensed articles that have been made previously available by Elsevier Labs.<sup>6</sup> These articles were the ba-

<sup>5</sup><https://github.com/harperco/MeasEval/tree/main/annotationGuidelines>

<sup>6</sup><https://github.com/elsevierlabs/OA-STM-Corpus>

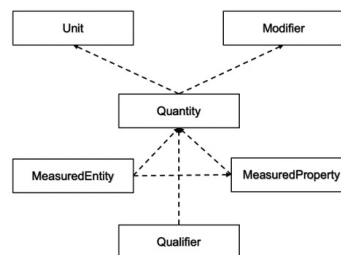


Figure 1: Annotation Model. All relationships are optional.

sis of a previous SemEval task for SemEval 2017 (Augenstein et al., 2017). These 110 articles are distributed evenly across 10 subject areas.

From these 110 articles, the MeasEval dataset includes 428 paragraphs containing 1663 Quantities. These are split into a training data set of 1164 Quantities (313 paragraphs) and an evaluation set of 499 Quantities (135 paragraphs).

All paragraphs were annotated by at least two annotators, then reviewed and reconciled during an adjudication meeting, often including a third annotator. The MeasEval data release included training data, as well as original annotations from multiple annotators for a 248 Quantity subset of the training data. This was to provide deep information on inter-annotator agreement, and also to allow participants to do their own analysis on how their algorithms perform relative to humans.

The inter-annotator agreement (IAA) shows some variation in interpretation when humans are performing this task. The review process serves to resolve much of the disagreement and to ensure that the data is as consistent as possible given the challenging nature of the task. For this subset of data in this IAA set, Table 1 shows Krippendorff’s Alpha values for each class.

Annotation Class	Krippendorff’s Alpha
Quantity	0.943
MeasuredProperty	0.641
MeasuredEntity	0.546
Qualifier	0.334
Unit	0.866

Table 1: Krippendorff’s Alpha scores for subset of data included in Inter-Annotator Agreement dataset.

### 4.3 Data Formats

To increase the usability of the data, multiple formats are provided. The MeasEval data includes a



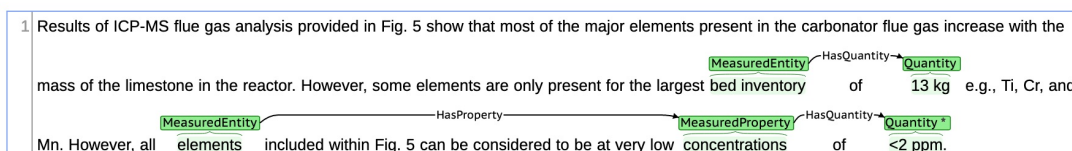


Figure 2: BRAT Example of a Quantity with related annotations.

text file and a set of annotations for each paragraph of scientific text. Annotations are provided in a tab-separated value (.tsv) file format, and in the BRAT annotation format. The BRAT format is for the purpose of visualization and review, but the official data format for the task is the .tsv, which is used for submissions and evaluation. For .tsv and .txt files, there is one file per paragraph of annotated text, and the .tsv file contains all annotations. For the BRAT files, there are one .ann and one .txt file per annotated Quantity.

For example, given the BRAT annotations illustrated in Figure 2, the data will have a raw text file (S0016236113008041-3153.txt), a BRAT annotation file *per Quantity* (S0016236113008041-3153-1.ann, and S0016236113008041-3153-2.ann), and a tab-separated file containing all annotations from each Quantity (S0016236113008041-3153.tsv).

More detail on each of these formats, including examples, as well as all MeasEval training and evaluation data, inter-annotator agreement annotations, and annotation guidelines can be found on the MeasEval Github repository.<sup>7</sup>

## 5 Evaluation

Evaluation is scored by providing a single SQuAD-style (Rajpurkar et al., 2016) F1 (Overlap) score for each submission, averaged across all nine components of the five subtasks. The 9 components are the Quantity, MeasuredProperty, MeasuredEntity, and Qualifier spans; the Modifier and Unit extensions to Quantity, and the HasQuantity, HasProperty, and Qualifies relationships. The evaluation script also provides a number of other metrics, described below.

In order to effectively evaluate all 9 components of the sub-tasks, it is necessary to first pin all Quantities in a submission to the corresponding Quantities in the gold data. As an example, consider the sentence “The dog weighed 25 pounds, while the average weight of the cats was 9 lbs.” We want to avoid crediting correct MeasuredEntities if asso-

ciated with the wrong Quantity. For example, if a submission listed “dog” as the MeasuredEntity associated with the average weight of 9 lbs, this would be incorrect.

The first pass matches each submission “annot-Set” ID to a corresponding Gold Set annotationId, and propagates this matched identifier across all of the data.

From there, the script calculates Precision, Recall, F-measure, and an Exact Match and SQuAD-style F1 (overlap) score. Exact Match and F1 are averaged across the entire submission. Exact Match is a binary value of 0 or 1, while F1 is a token level overlap ratio of submission to gold spans, where tokenization is done using simple white space delimiters. For components that do not include a span, Exact Match and F1 scores are the same. Relations are also scored with a binary Exact Match and F1 score if the relation types match and both endpoints match either exactly or with some overlap.

Any span, unit, modifier, or relationship found in the gold data, but not the submission, or found in the submission, but not the gold data is included as a “penalty row” with a score of 0 in order to sufficiently penalize both false positives and false negatives when averaging scores. This calculation leads to very fine-grained differences in the distribution of scores in the results tables.

Although not used for calculating leaderboard rankings, the evaluation code can also provide all the same scores micro-averaged by scoring component, by subject area, or by paragraph for further analysis of error. Additional documentation as well as the evaluation code itself can be found on the MeasEval GitHub repository.<sup>8</sup>

### 5.1 Baseline Models

MeasEval also includes two very similar baseline models. Baseline 1 is the best-performing of these, and scores an overall F1 (Overlap) of 0.239 in the evaluation as reported in Tables 2 and 3. Base-

<sup>7</sup><https://github.com/harperco/MeasEval>

<sup>8</sup><https://github.com/harperco/MeasEval/tree/main/eval>

Team Name	Overall	Quantity	Unit	Modifier	MeasuredEntity	MeasuredProperty	Qualifier
LIORI*	<b>0.519</b>	<b>0.861</b>	0.722	<b>0.642</b>	<b>0.437</b>	<b>0.467</b>	<b>0.163</b>
jarvis@tencent*	<u>0.473</u>	<u>0.855</u>	0.719	0.523	<i>0.398</i>	<u>0.437</u>	0.000
zzy_77	<i>0.448</i>	0.842	0.697	0.507	0.383	<u>0.385</u>	0.000
zz362	0.433	0.821	0.720	0.498	0.344	0.365	0.000
Counts@IITK*	0.432	<b>0.861</b>	<u>0.406</u>	0.245	<i>0.077</i>	<b>0.804</b>	<u>0.614</u>
yorkey	0.399	0.745	0.661	0.314	0.344	0.365	0.000
XMSHI	0.392	0.736	0.624	0.313	0.348	0.353	0.000
CLaC-BP*	0.389	<u>0.855</u>	0.677	<i>0.546</i>	0.251	0.318	<u>0.107</u>
clockwise9*	0.369	<i>0.850</i>	0.618	0.000	0.327	0.350	0.000
UPB*	0.369	0.742	0.533	0.277	0.331	0.374	0.040
<i>Baseline</i>	0.239	0.827	0.561	0.000	0.053	0.064	0.005
KGP*	0.278	0.787	<i>0.748</i>	0.309	0.113	0.012	0.005
Stanford MLab*	0.272	0.818	<u>0.760</u>	0.408	0.000	0.000	0.000
BuckschJ	0.263	0.825	0.695	0.375	0.000	0.000	0.000
CLaC-np*	0.241	0.756	0.495	0.408	0.056	0.006	0.000
FabianW	0.238	0.826	0.624	0.438	0.060	0.045	0.006
ugeijtsv	0.229	0.759	0.582	0.210	0.000	0.000	0.000
Jo	0.212	0.754	0.377	0.291	0.000	0.000	0.000
joe.o123	0.185	0.376	0.383	0.242	0.000	0.000	0.000
SU-NLP	0.001	0.007	0.002	0.000	0.000	0.000	0.000

Table 2: Top result for each team/user, ordered by Overall F1 along with micro-averages for each annotation span, for units, and for modifiers. Team Names marked with \* have submitted system information for further analysis and discussion. Top, second, and third place scores per category represented by **bold**, underline, and *italics* respectively.

line 1 use spaCy Named Entity Recognition (NER) models for each of the four classes independently. Unfortunately, some training examples need to be thrown away because spaCy’s NER functionality does not support overlapping spans in the same model. Since there is frequently an overlap between MeasEval spans of different types, this necessitates training each annotation type separately, and stripping out edge cases where multiple annotations of the same type intersect.

Baseline 1 generates a deduplicated list of all units in the training data, and checks each Quantity against this list. If there are one or more matches in this comparison, the system returns the “longest last matching” unit, ensuring that cm would be preferred to m in “22 cm” and that s would be preferred to m in “approximately 22 s”. The baseline does not attempt the Modifier component, though could be augmented with a set of regular expressions that search the Quantity string for key phrases and symbols, including “approximately”, “between”, “>”, and “~”.

Once the NER models and unit matching are completed, baseline 1 matches Quantities to MeasuredEntities, MeasuredProperties, and Qualifiers using a knockout match algorithm based on proximity. So each MeasuredProperty matches the nearest Quantity, each MeasuredEntity matches the nearest MeasuredProperty or Quantity, and each Qualifier

matches the nearest span of any type. Baseline 2 is a variant that does much simpler matching, taking each span in the order they appear in the data. Baseline 2 does not appear in the results tables, but scores an overall F1 (Overlap) of 0.223. The code for both baselines is available in a Jupyter notebook on the MeasEval Github repository.<sup>9</sup>

## 6 Results and Discussion

During the 21-day evaluation period (January 10 through 31, 2021), 26 CodaLab users submitted a total of 89 submissions, of which 77 passed validation and were successfully scored by the evaluation script. Given the complexity of the task, we opted to allow for five submissions total during the evaluation, although some collaboration between users meant that some teams were able to effectively submit more than five times. We note that submissions did not calculate scores on sub-tasks, thus making it difficult to overly optimize models using just the overall score. The relatively generous submission allowance does not seem to have presented too much of an over-fitting problem, as scores remain relatively low on all tasks, although the collaboration could have given some participants a slight advantage in the rankings.

Table 2 shows the top submission from each of

<sup>9</sup><https://github.com/harperco/MeasEval/blob/main/baselines/first-baseline.ipynb>

the 19 teams that submitted successfully, as well as the top performing baseline. 10 of the 19 exceed the benchmark of the baseline spaCy model. In addition to the overall F1 scores, Table 2 shows micro-averaged F1 across the four annotation spans as well as Units and Modifiers. Table 3 provides this same breakdown for each of the three relationship types. Team names marked with an asterisk (\*) represent teams which have either submitted system description papers or responded to a request for system information.

The overall top-performing model was at least tied for top performance in five out of 6 of the component scores in Table 2, but interestingly, the second best and third best performing models varied across scoring component. Models that did particularly well at Quantities, Units, or Modifiers, may have had their overall performance reduced by lower performance at the MeasuredEntity and MeasuredProperty spans.

Table 3 shows the scores for the Relation Extraction subtasks: HasQuantity, HasProperty, and Qualifies. These largely align with the annotation span components of the scoring which they are dependent on. In both Table 2 and Table 3 it is worth noting that only 7 teams attempted extraction of Qualifiers and the Qualifies relation, as these were the most difficult aspects of the task.

Team Name	HasQuantity	HasProperty	Qualifies
LIORI*	<b>0.482</b>	<b>0.318</b>	<b>0.092</b>
jarvis@tencent*	<u>0.424</u>	<u>0.257</u>	0.000
zyy_77	<i>0.387</i>	0.229	0.000
zz362	0.375	0.203	0.000
Counts@IITK*	0.311	0.183	<u>0.064</u>
yorkey	0.375	0.203	0.000
XMSHI	0.373	0.199	0.000
CLaC-BP*	0.308	0.147	<i>0.058</i>
clockwise9*	0.366	0.167	0.000
UPB*	0.350	<i>0.242</i>	0.019
<i>Baseline</i>	0.075	0.007	0.000
KGP*	0.076	0.006	0.000
Stanford MLab*	0.000	0.000	0.000
BuckschJ	0.000	0.000	0.000
CLaC-np*	0.028	0.000	0.000
FabianW	0.037	0.007	0.000
ugeijtsv	0.000	0.000	0.000
Jo	0.000	0.000	0.000
joe_o123	0.000	0.000	0.000
SU-NLP	0.000	0.000	0.000

Table 3: Top result for each team/user for the Relation Extraction components of the score. Team Names marked with \* have submitted system information for further analysis and discussion. Top, second, and third place scores per category represented by **bold**, underline, and *italics* respectively.

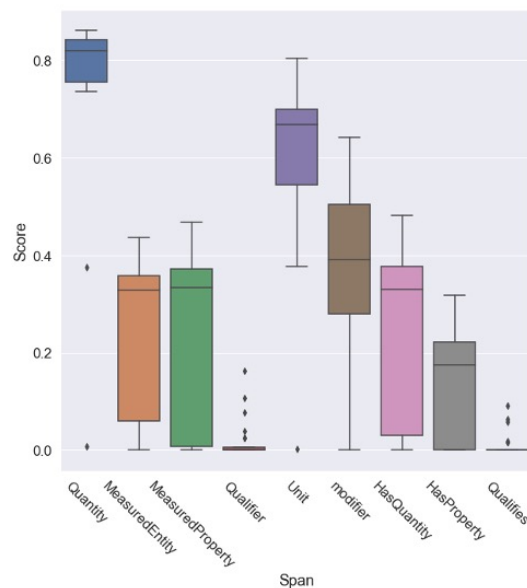


Figure 3: Visualization of average scores for each scoring component across top score for all participants.

Figure 3 provides a visualization of the distribution of scores for each scoring component from Tables 2 and 3. From this, it is clear that Quantity and Unit are the easiest aspects of the shared task, which makes intuitive sense. The relatively high scores for Modifier is also of interest, as these are the components of the extraction that capture uncertainty and variance in value, which is an important part of the task and not one that we expected to see handled as well as it was. This clearly demonstrates that the various Quantity contextualization subtasks are far more difficult and more work is needed in how best to handle the extraction of related MeasuredEntities, MeasuredProperties, and Qualifiers.

Figure 4 provides a visualization of the distribution of scores for 9 of the 10 subject areas in the corpus. The mathematics subject area has been omitted from this graphic due to under-representation in both the training and evaluation datasets.

## 6.1 Impact of Duplicates

As noted previously, the MeasEval evaluation algorithm was designed toward lenience, and as a result sometimes inflates scores if multiple submission annotations match a single entry in the gold data. This was done to allow submissions to get credit for submitting multiple Quantity annotations that partially matched a single gold data span as well as to generally not penalize systems that might make multiple predictions pinned to the same numeric

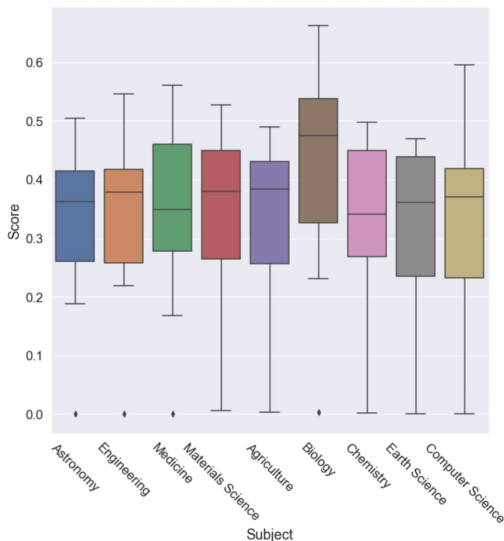


Figure 4: Visualization of average scores for each subject area across top score for all participants.

value.

However, allowing duplicates can in some cases result in inflated scores. This is especially evident in cases where submissions contained entries that duplicated entire annotations sets. For a small number of submissions that exhibited annotation set level duplicates, a post-processing routine removed all set level duplicates before final evaluation, resulting in the scores in Table 2 and Table 3.

Additionally, Quantity-level duplicates can also potentially inflate Quantity scores, but should have a neutral impact on other components of scoring. For example, if a system identified the same Quantity two times, but found a different MeasuredEntity for each occurrence, the submission will score extra points associated with the Quantity, and potentially the Unit and Modifiers if those are also correct, but will only get points for the correct MeasuredEntity while being penalized for the incorrect MeasuredEntity. An ablation analysis was performed for the eight submissions covered by system papers, assessing the impact of these duplicates on the Overall F1 (Overlap) metric.

Table 4 gives the extent of duplication for these submissions, the initial overall score from Table 3, the overall score with exact quantity duplicates removed, and the overall score with both exact and overlapping duplicates removed. For example, if the gold data includes the Quantity “approximately 23 mm”, and a submissions included annotation sets with both “23 mm” and “approximately 23 mm”, the exact match duplicate removal would not

drop either score, whereas the overlapping match score would drop whichever occurred last in the submission, whether or not it is the correct answer.

Team Name	F1	Count Exact / Overlap	F1 w/out Exact	F1 w/out Any Duplicates
LIORI	0.519	125 / 32	0.499	0.487
jarvis@tencent	0.473	0 / 11	0.473	0.470
Counts@IITK	0.432	0 / 0	0.432	0.432
CLaC-BP	0.389	0 / 0	0.389	0.389
UPB	0.369	0 / 1	0.369	0.369
KGP	0.278	0 / 0	0.278	0.278
Stanford MLab	0.272	0 / 0	0.272	0.272
CLaC-np	0.241	55 / 0	0.231	0.231

Table 4: Ablation analysis of duplicates and Overall F1 (Overlap) score for each of the eight Teams with System Papers.

The general downward trajectory seen while removing duplicates that are not at the set level is informative. Partly this is due to declining Quantity score from duplicate removal, but some effect is attributable to the possibility that deduplication deletes a correct MeasuredProperty or MeasuredEntity and leaves an incorrect one, given that they may include different values. The ablation analysis simply removes all but the first occurrence, so there is no control over whether removed values are a closer match to the gold data.

## 6.2 Multiple Hypotheses Hypothesis

The results of de-duplication analysis, the relatively low inter-annotator agreement scores, and deeper consideration of the annotation guidelines present an interesting hypothesis. It could be that the different interpretations of the context of a measurement are not automatically right or wrong. It could be that different interpretations are useful in different downstream applications. While it is out of scope in for this task description, future work may look more closely at categorization of the areas where annotators disagreed and systems produced multiple interpretations, to see if there is alignment in the differences and whether there are patterns to the variance.

## 7 Summary of Participating Systems

The MeasEval track at SemEval-2021 received nine system description paper submissions, eight of which are represented in the analysis in Table 4. A ninth paper formulated a new task from the MeasEval dataset focusing on just the relation extraction

part of the problem. One system only attempted the Quantity, Unit, and Modifier parts of the task, while another did not submit any Qualifier spans or Qualifies relationships. Four of the nine systems have released code or models.

There are several points of similarity between the eight main submissions. All but one of the systems are based on the BERT architecture (Devlin et al., 2018) or a derivative, such as SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2019), or RoBERTa (Liu et al., 2019). All but one used a pipeline architecture, starting with Quantity extraction. All but one used sequence tagging with a BIO encoding scheme, and four followed the sequence tagger by a Conditional Random Fields (CRF) model to assemble tokens into spans and improve accuracy over simple token-level classifiers. Unit and Modifier extraction was either done using a character-level BiLSTMs, another BERT model, or a rule-based approach. Finally, it was common to see MeasuredEntity, MeasuredProperty, and Qualifier, and sometimes the relation extraction components, stacked together in a multi-task sequence tagging model as a final stage, taking both the original sentences and Quantity spans as input. One system diverged from the sequence tagging approach and used templated question answering techniques to handle the relation extraction along with related spans. Table 5 provides a high-level summary of frequency of architectures and techniques in use by more than one system.

Technique / Model	Submission Count
BERT	3
BioBERT	1
SciBERT	3
RoBERTa	1
CRF	4
BiLSTM Units / Mods	3
Rule-based Units / Mods	3
Dict-based Units / Mods	2
Question Answering	1
Sequence Tagging	7

Table 5: Summary of techniques and architectures used in MeasEval System Description Submissions.

## 7.1 System Specifics

Davletov et al. (2021) (LIORI), achieve their state-of-the-art performance using pre-trained models RoBERTa (Liu et al., 2019) and LUKE (Yamada et al., 2020). They use LUKE to fine-tune an NER model for Quantity extraction, and a RoBERTa-based multi-task model for all other spans. Mod-

ifiers are predicted as Quantity-types. All other spans, including units, are extracted using Question Answering style sequence tagging (start/end logits) without question prompts for each annotation type queried for each extracted Quantity.

This sequential ensemble approach of Quantity-detection followed by either “all-in-one-multi-task” extraction or a staged approach to one or more of the other subtasks proved very common among the top-performing systems.

Cao et al. (2021) (jarvis@tencent), do initial Quantity extraction with an ensemble of a Pointer Net (Vinyals et al., 2015) and a CRF. They use a BERT-based classifier for Modifier tagging and a rule-based system for Units, and then use relation-specific taggers with the same architecture as the Quantity-tagger for all other task components.

Gangwar et al. (2021) (Counts@IITK) similarly tag Quantities with a SciBERT sequence tagger and a CRF model and SciBERT for Modifiers, but use a Character based bidirectional LSTM for Unit tagging. They then encode the Quantity into SciBERT input when tagging MeasuredEntity and HasQuantity, and iteratively mark new spans in the input when tagging then next sub-task, using a rule-base for assembling the necessary relationships. Their performance on Quantity, Unit, and Modifier was near the top performing, but they struggled with MeasuredProperty and HasProperty.

Therien et al. (2021) (CLaC-BP) use SciBERT in a token-level multi-class classifier across all span classes. This is an interesting approach, given the opportunity for joint inference between the various types of spans. However, it penalizes them in that each token in their model can only be one class, while there are cases when a Quantity and MeasuredEntity from one set may be part of, e.g. a Qualifier in another. Quantity spans are then fed to another SciBERT model for Modifier typing, and rule-based systems are used for Units and for the Relations between spans.

Avram et al. (2021) (UPB) use RoBERTa along with a CRF for Quantity extraction. They also tested SciBERT and BERT. They achieved their best results on their dev subset with SciBERT, but their best results on evaluation set came from RoBERTa. They use a BiLSTM to extract Units and classifier Modifiers, and then use a templated Question Answering system as a joint entity and relation extraction system for the remaining subtasks. Unlike LIORI, who did not use prefixes or

suffixes in their question templates, UPB asks more natural language questions of the form “What is the measured property of the quantity \_\_\_?”

[Karia et al. \(2021\)](#) (KGP) also use BioBERT after testing various BERT-based pre-trained models, but depart from many of the other submissions by using a binary classifier rather than BIO tags and CRF layers for Quantity sequence tagging. Modifiers and Units are handled using keywords and dictionary matching, while they use a multi-task BERT model for the remaining components, first finding MeasuredEntity based on the Quantity predictions, then fusing these results for the remaining spans and relations. They also continued refining their approach into the post-evaluation phase, and reported improving their score from an Overall F1 (Overlap) of 0.278 to 0.456.

[Liu et al. \(2021\)](#) (Stanford MLab) only tackle the Quantity, Unit, and Modifier subtasks. Notably, they report building their system for these components from inception to submission in under 48 hours. They use BERT-large for IOB sequence tagging for Quantities, use a similar IO sequence tagging scheme on Quantities to tag Units, and a multi-class classifier to classify Quantities to the appropriate Modifiers. Their system performs well on all subtasks they attempted, even scoring second place overall for Units.

[Lathiff et al. \(2021\)](#) (CLaC-np) diverge from other submissions in their approach. They preprocess their text using GATE and ANNIE, and use custom rules to further clean up tokenization. They treated Stanford Core Dependency Parse trees as graphs to extract subgraphs starting each path query from the CD tokens to identify MeasureEntities, MeasuredProperties and Qualifiers with the use of Graph CNN. They relied on the models from CLaC-BP to map from their tokens to annotation spans for each type in assembling their final submission.

Finally, not shown in Table 1 is [Veyseh et al. \(2021\)](#). They formulated their own task based on the MeasEval data. Although they did not submit a solution during the evaluation period, they have submitted a system description paper describing a novel approach to relation extraction, which they have evaluated on MeasEval sub-task 5. Using our Gold Quantity, MeasuredEntity, MeasuredProperty, and Qualifier spans as input (without annotation sets), they compared their approach to two other baseline models. They encode contextual embeddings, positional embeddings, and entity types

for each annotation span, and perform dependency path reasoning along with an “Information Bottleneck” regularization technique to complete their Relation Extraction task.

## 8 Conclusion

In this paper, we present the design, the data, the evaluation the process, the results, and the systems for MeasEval at SemEval 2021. The shared task is challenging, partly due to the relatively small training data, and partly due to the inter-relationships between many different components of the task. Quantity and Unit identification, and to a lesser extent Modifier typing, appear to be the simplest parts of the task based on average performance, with one participating system building their end-to-end pipeline for these components in under 48 hours. The contextual elements MeasuredEntity, MeasuredProperty, and Qualifier, and their relationships, are far more difficult, which is not surprising given that these are subject to more human annotator disagreement. The challenge of context is especially pronounced in the Qualifier span and Qualifies relationship.

Common components shown in Table 5 include the BERT family of pre-trained neural language models, CRF models, BiLSTMs, and rule-based components. In general, the task does not appear to require whole new novel models and architectures, but rather pipelines and cascading ensembles stitching together various existing methods. There is still room for improvement on this task, and whether progress will come from novel models or creative applications of existing techniques remains to be seen. Work also remains to be done in applying the entities and relationships extracted for this task to the larger end goal of scientific knowledge graph construction and related downstream applications. Future work could be done to further analyze areas of error and disagreement in these annotations, and to investigate entity linking across Quantity, MeasuredEntity, and MeasuredProperty annotation spans to various measurement ontologies and to domain-specific entity and property ontologies.

## Acknowledgments

The authors would like to thank Darin McBeath and Pierre-Yves Vandenbussche for help with annotations and annotation rules. We also thank Elsevier’s Discovery Lab team for their feedback on this work.

## References

- Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. [SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.
- Andrei-marius Avram, George-eduard Zaharia, Dumitru-clementin Cercel, and Mihai Dascalu. 2021. UPB at SemEval-2021 Task 8 : Extracting Semantic Information on Measurements as Multi-Turn Question Answering. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, 2.
- Payam Barnaghi, Wei Wang, Lijun Dong, and Chong-gang Wang. 2013. A linked-data model for semantic sensor streams. *Proceedings - 2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing, GreenCom-iThings-CPSCom 2013*, pages 468–475.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A Pretrained Language Model for Scientific Text](#). pages 3613–3618.
- Melanie Bergmann, Mine B Tekman, and L. Gutow. 2017. [LITTERBASE: An Online Portal for Marine Litter and Microplastics and Their Implications for Marine Life](#), pages 106–107.
- Jiarun Cao, Yuejia Xiang, Yunyan Zhang, Zhiyuan Qi, and Xi Chen. 2021. CONNER : A Cascade Count and Measurement Extraction Tool for Scientific Discourse. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Dmitry Davidov and Ari Rappoport. 2010. Extraction and approximation of numerical attributes from the Web. *ACL 2010 - 48th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, (July):1308–1317.
- Adis Davletov, Denis Gordeev, Nikolay Arefyev, and Emil Davletov. 2021. LIORI at SemEval-2021 Task 8: Ask Transformer for measurements. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Helena F. Deus, Corey A. Harper, Darin McBeath, and Ron Daniel. 2017. Combining pattern matching with word embeddings for the extraction of experimental variables from scientific literature. *2017 IEEE International Conference on Big Data (Big Data)*, pages 4287–4292.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#).
- Chau Do and Eric J. Pauwels. 2013. Using MathML to represent units of measurement for improved ontology alignment. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7961 LNAI:310–325.
- Liran Einav and Jonathan Levin. 2014. [The data revolution and economic analysis](#). *Innovation Policy and the Economy*, 14:1–24.
- Luca Foppiano, Thaer M Dieb, Akira Suzuki, and Masashi Ishii. 2019a. Proposal for Automatic Extraction Framework of Superconductors Related Information from Scientific Literature Proposal for Automatic Extraction Framework of Superconductors Related Information from Scientific Literature. (June):0–5.
- Luca Foppiano, Laurent Romary, Masashi Ishii, and Mikiko Tanifuji. 2019b. [Automatic identification and normalisation of physical measurements in scientific literature](#). *Proceedings of the ACM Symposium on Document Engineering, DocEng 2019*, pages 0–4.
- Akash Gangwar, Sabhay Jain, Shubham Sourav, and Ashutosh Modi. 2021. Counts @ IITK at SemEval-2021 Task 8 : SciBERT Based Entity And Semantic Relation Extraction For Scientific Data. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Abhijeet Gupta, Gemma Boleda, Marco Baroni, and Sebastian Padó. 2015. [Distributional vectors encode referential attributes](#). *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, (September):12–21.
- Tianyong Hao, Hongfang Liu, and Chunhua Weng. 2016. [Valx: A system for extracting and structuring numeric lab test comparison statements from text](#). *Methods of information in medicine*, 55.
- Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. [Knowledge graphs](#).
- Kyle Hundman and Chris A Maamann. 2017. [Measurement Context Ex-traction from Text: Discovering Opportunities and Gaps in Earth Science](#). 8:pages.
- Joint Committee for Guides in Metrology. 2012. *International vocabulary of metrology. Basic and general concepts and associated terms.*, 3rd ed edition.
- Jocelyn Kaiser. 2018. [Plan to replicate 50 high-impact cancer papers shrinks to just 18](#). *Science*.

- Ramar Kaladevi, Gurunathan Geetha, and P. Narayanasamy. 2016. [Ontological based interoperability and integration framework for heterogeneous weather systems](#). *Revista Tecnica de la Facultad de Ingenieria Universidad del Zulia*, 39(1):185–192.
- Neel Karia, Ayush Kaushal, and Faraaz Rahman Mallick. 2021. [KGP at SemEval-2021 Task 8 : Leveraging Multi-Staged Language Models for Extracting Measurements, their Attributes and Relations](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Natalia Konstantinova. 2014. [Review of relation extraction methods: What is new out there?](#) *Communications in Computer and Information Science*, 436:15–28.
- Bhushan Kotnis and Alberto Garcia-Duran. 2019. [Learning Numerical Attributes in Knowledge Bases](#). *Automated Knowledge Base Construction (2019)*.
- Nihatha Lathiff, Pavel Khloponin, and Sabine Bergler. 2021. [CLaC-np at SemEval-2021 Task 8 : Dependency DGCNN](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. [BioBERT: a pre-trained biomedical language representation model for biomedical text mining](#). pages 1–8.
- Patrick Liu, Niveditha S Iyer, Erik Rozi, and Ethan A Chi. 2021. [Stanford MLab at SemEval-2021 Task 8 : 48 Hours Is All You Need](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *arXiv*, (1).
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#). (ii).
- H. Rijgersberg, M. Wigham, and J. L. Top. 2011. [How semantics can improve engineering processes: A case of units of measure and quantities](#). *Advanced Engineering Informatics*, 25(2):276–287.
- Hajo Rijgersberg, Mark Van Assem, and Jan Top. 2013. [Ontology of units of measure and related concepts](#). *Semantic Web*, 4(1):3–13.
- J. Sprague. 2006. [The Zebrafish Information Network: the zebrafish model organism database](#). *Nucleic Acids Research*, 34(90001):D581–D585.
- Markus D. Steinberg, Sirko Schindler, and Jan Martin Keil. 2017. [Use cases and suitability metrics for unit ontologies](#). *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10161 LNCS:40–54.
- Benjamin Therien, Parsa Bagherzadeh, and Sabine Bergler. 2021. [CLaC-BP at SemEval-2021 Task 8 : SciBERT Plus Rules for MeasEval](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Shreejoy J. Tripathy, Judith Savitskaya, Shawn D. Burton, Nathaniel N. Urban, and Richard C. Gerkin. 2014. [NeuroElectro: A window to the world’s neuron electrophysiology data](#). *Frontiers in Neuroinformatics*, 8(APR):1–11.
- Amir Pouran Ben Veyseh, Franck Deroncourt, and Thien Huu Nguyen. 2021. [DPR at SemEval-2021 Task 8: Dynamic Path Reasoning for Measurement Relation Extraction](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. [Pointer networks](#). *Advances in Neural Information Processing Systems*, 2015-January:2692–2700.
- Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. [LUKE: Deep contextualized entity representations with entity-aware self-attention](#). *arXiv*, pages 6442–6454.



# SemEval-2021 Task 9: Fact Verification and Evidence Finding for Tabular Data in Scientific Documents (SEM-TAB-FACTS)

Nancy X. R. Wang\* Diwakar Mahajan\* Marina Danilevsky Sara Rosenthal§  
IBM Research  
nancywang1991@gmail.com, {dmahaja, mdanile, sjrosenthal}@us.ibm.com

## Abstract

Understanding tables is an important and relevant task that involves understanding table structure as well as being able to compare and contrast information within cells. In this paper, we address this challenge by presenting a new dataset and tasks that addresses this goal in a shared task in SemEval 2020 Task 9: Fact Verification and Evidence Finding for Tabular Data in Scientific Documents (SEM-TAB-FACTS). Our dataset contains 981 manually-generated tables and an auto-generated dataset of 1980 tables providing over 180K statement and over 16M evidence annotations. SEM-TAB-FACTS featured two sub-tasks. In sub-task A, the goal was to determine if a statement is supported, refuted or unknown in relation to a table. In sub-task B, the focus was on identifying the specific cells of a table that provide evidence for the statement. 69 teams signed up to participate in the task with 19 successful submissions to subtask A and 12 successful submissions to subtask B. We present our results and main findings from the competition.

## 1 Introduction

Tables are ubiquitous in documents and presentations for conveying important information in a concise manner. This is true in many domains, stretching from scientific to government documents. In fact, surrounding text in these articles are often statements summarizing or highlighting some information derived from the primary source of data in tables. A relevant example is shown in Figure 1 from a Business Insider article analyzing the impact of Covid-19 (Aylin Woodward and Gal, 2020). Describing all the information provided in this table in a readable manner would be lengthy and considerably more difficult to understand. Despite their importance, popular question answering (e.g. SQuAD and Natural Question (Rajpurkar et al.,

\* Equal Contribution

§ Corresponding Author

**The total number of cases and deaths have far surpassed those of the SARS outbreak.**

### 2019 novel coronavirus compared to other major viruses

VIRUS	YEAR IDENTIFIED	CASES	DEATHS	FATALITY RATE	NUMBER OF COUNTRIES
Ebola	1976	33,577	13,562	40.4%	9
Nipah	1998	513	398	77.6%	2
SARS	2002	8,096	774	9.6%	29
MERS*	2012	2,494	858	34.4%	28
COVID-19**	2020	222,642	9,115	4.1%	159

Sources: Johns Hopkins, CDC, World Health Organization, New England Journal of Medicine, Malaysian Journal of Pathology, CGTN

\*As of November 2019 \*\*As of March 19, 2020 at 7:30 am EST.

BUSINESS INSIDER

Figure 1: Surrounding text often highlights some information from the table but does not capture all data. Alternately, the linked text may be subjective or even misleading without the original table to check the claims.

2016; Kwiatkowski et al., 2019)) and truth verification tasks (e.g. SemEval-2019 Fact Checking Task (Mihaylova et al., 2019)) have not focused on tables, being composed solely of written text. This is likely due to their complexity to parse and understand, despite their rich amount of information.

Further, the structure of tables allows much more information to be presented in an efficient manner as humans can interpret meaning in the spatial relationship between cells. However, due to their challenging nature, recent algorithms have been less successful at extracting (Hoffswell and Liu, 2019) and understanding header and data structure in tables (Cafarella et al., 2018). In addition, any hierarchical and nested headers (common in printed documents) increases the difficulty in interpreting data cells, as shown in Figure 2.

In this paper, we propose to bridge this gap with statement verification and evidence finding using tables from scientific articles. This important task promotes proper interpretation of the surrounding article. In fact, the misunderstanding of tables can lead to the reporting of fake news that we see as being all too prevalent today.

	n (% initiated smoking)	Unadjusted	
		OR (95% CI)	p
Baseline EC use			
Never	902 (8.2)	1.00	
Ever	21 (52.6)	12.41 (4.53–33.99)	<.001
Follow-up EC use			
No escalation	882 (8.1)	1.00	
Escalation	41 (41.0)	7.94 (3.75–16.82)	<.001
Age			
11–13	397 (4.4)	1.00	
14–15	270 (6.3)	1.45 (.71–2.97)	.312
16–18	256 (16.1)	4.12 (2.19–7.76)	<.001

Figure 2: A complex table sourced from (East et al., 2018) with hierarchical column and row structure. Additional difficulty follows from row hierarchy not being delineated by separate columns.

We present the first SemEval challenge to address table understanding. We introduce a brand new dataset of 1980 tables from scientific articles that addresses two challenging tasks important to table understanding:

**A: Statement Fact Verification** Given a statement, determine whether it is *supported*, *refuted* or *unknown* according to the table.

**B: Cell Evidence Selection** Given a statement, select the cells in the table that provide evidence supporting or refuting the statement.

The rest of this paper is formatted as follows: We first discuss related work. We then present a new large table understanding dataset containing close to 2000 tables that is the first to provide evidence labels at the cell level for statements and the first to focus on scientific articles. We provide a detailed analysis of the dataset including several baseline results. We then discuss the performance and approaches of the 19 participants in our challenge and end with an aggregated analysis of participating teams. Finally, we discuss future work.

## 2 Related Work

**Natural Language Inference (NLI)** The table evidence task can be best understood as a variation of the natural language inference task (Dagan et al., 2005), but on tabular data. NLI asks whether one (or more) sentence entails, refutes, or is unrelated to another sentence; our framing asks whether a given table entails, refutes, or is unrelated to a sentence. Several datasets have been created for studying NLI, such as SNLI (Bowman et al., 2015), MultiNLI (Williams et al., 2018), and SciTail (Khot et al., 2018).

**Table QA** This task is also closely related to the problem of search and question answering on ta-

bles. The closest example would be, given a table that is known to contain the relevant information, return cell values that answer a natural language question (Pasupat and Liang, 2015). A variation requires analyzing a collection of tables rather than a single one, along with the natural language question (Sun et al., 2016). Two of the most recent works are TAPAS (Herzig et al., 2020) and TaBERT (Yin et al., 2020), which jointly pre-train over textual and tabular data to facilitate table QA. However, such approaches have previously focused on traditional natural language questions (“What is the population of France?”) rather than inference statements (“France has the highest population in Europe”), which may be entailed, refuted or unknowable from the given table.

**Related Datasets** The works closest to our dataset are TabFact (Wenhu Chen and Wang, 2020) and INFOTABS (Gupta et al., 2020). Both datasets were sourced from Wikipedia tables and contain hypothesis and premise pairs. TabFact has entailment and refute hypothesis types while INFOTABS has an additional “neutral” hypothesis category, much like our “unknown” statements. Both works show that neural models still lag far behind human performance for the fact checking task with tables.

While both datasets have been great at kindling interest in fact verification with tabular data, our dataset differs in two key aspects. First, we source from scientific articles in a variety of domains rather than Wikipedia infoboxes. Scientific tables have very specialized vocabulary and can be more difficult to interpret. Additionally, scientific tables have much more complex structure, like hierarchical column and row headers, rendering the assumption that the first column/row is the header unhelpful. Finally, tables are often directly referenced in scientific text unlike Wikipedia tables that are generally stand-alone. This creates an opportunity to leverage natural statements that depict the original author’s style and intent. The second key differentiator of SEM-TAB-FACTS is the accompanying evidence annotations. We believe the future of fact verification and AI in general will be in cooperation with humans rather than in replacement. Thus, it is essential that models are able to present explanations for decisions on the relationship between the statement and table by showing the most relevant cells in a potentially very large table.

Source	#Tables	#Entailed	#Refuted	#Unknown	#Relevant	#Irrelevant
Train Crowdsourced	981	2,818	1,688	0	0	0
Train Auto-generated	1,980	92,136	87,209	0	1,039,058	15,467,957
Development	52	250	213	93	3,048	2,8495
Test	52	274	248	131	3,458	26,724

Table 1: Statistics for our SEM-TAB-FACTS dataset.

### 3 Dataset Details

Our dataset consists of two forms of generation: (1) a crowdsourced dataset, and (2) an auto-generated dataset. Table 1 presents the statistics of the dataset. We detail our dataset creation process in the following sections.

#### 3.1 Data extraction and preprocessing

We sourced our tables from scientific articles belonging to active journals that are currently being published by Elsevier and are available on ScienceDirect<sup>1</sup>. We utilized Elsevier ScienceDirect APIs<sup>2</sup> to scrape scientific articles which belong to this list, and satisfy the following criteria: (1) the article is open-access<sup>3</sup>, (2) the article is available under “Creative Commons Attribution 4.0 (CC-BY)” user license<sup>4</sup>, and (3) the article has at least one table. We downloaded 1,920 articles belonging to 722 journals which contained 6,773 tables. We further filtered out complicated tables (e.g. multiple tables in a single table) using hand-written rules to get a set of 2,762 candidate tables from 1,085 articles for annotation. We also extracted sentences mentioning the table within the scientific article as candidate statements, which are corrected and then labeled manually by the annotators.

#### 3.2 Crowdsourced labeling

The manually generated statements were collected using the crowdsourcing platform Appen<sup>5</sup>. We collected five entailed and five refuted statements for each table from the business preferred operators (BPO) on Appen. The BPO crowd is composed of employees hired by Appen on an hourly basis at a constant pay rate determined by Appen. We found

that the workers were much more motivated for the task as they were able to ask questions if needed and we were also able to provide direct feedback to the workers. We initially attempted generating statements with workers from the Appen open-crowd, which is on-demand, but the quality was very poor as it was hard to automatically validate naturally generated statements. Our instructions explicitly lay out 7 types of statements and ask that workers attempt to make one of each type. We encourage the use of different sets of cells whenever possible. The types of statements are aggregation, superlative, count, comparative, unique, all and usage of caption or common sense knowledge. These are derived from the INFOTABS analysis (Gupta et al., 2020). We asked workers to avoid subjective adjectives like “best”, “worst”, “seldom” and look-up statements that only require reasoning with one cell. The pay for each statement set was 75 cents. In total, we collected 10000 statements for 1000 unique tables. See Figure 3 for an example table with its manually generated and natural statements.

Additionally, for our training data, we conducted a verification task to check for grammatical issues and doubly verify the statement label for both the generated and natural in-text statements. The verification task was paid at 3 cents per statement, which equates to 30 cents per table. We restricted the verification task to the workers in the open-crowd from English speaking countries. After verification, we only preserved the statements that were verified to be grammatically correct and the new label matched the original label. Natural statements were also verified in the same process. Although natural statements were generally factually correct, they were sometimes not able to be verified by the referenced table. Additionally, these statements often required rewording to ensure that all parts of the statement can be verified by the table, which was a step taken only for the development and test sets. This left us with 981 tables and 4506 statements. The majority of the removals were due to

<sup>1</sup>[https://www.elsevier.com/\\_\\_data/promis\\_misc/sd-content/journals/jnlactive.xlsx](https://www.elsevier.com/__data/promis_misc/sd-content/journals/jnlactive.xlsx)

<sup>2</sup>[https://dev.elsevier.com/sd\\_apis.html](https://dev.elsevier.com/sd_apis.html)

<sup>3</sup><https://www.elsevier.com/open-access>

<sup>4</sup><https://www.elsevier.com/about/policies/open-access-licenses/user-licences>

<sup>5</sup><https://appen.com/>

Table 2

Data are for 1290 firms across nine East Asian economies. All network data are assembled by the authors, and are cross-sectional for 2008. Table reports country-level statistics on board networks, family networks, state networks, and political networks. Minimum values are everywhere 0. board network counts the amount of board/executive interlocks. Political network counts the amount of board/executive interlocks with politically-connected firms. Family network counts the amount of board/executive interlocks with family-controlled firms. State network counts the amount of board/executive interlocks with state-owned firms.

Country	N	Board network			Family network			State network			Political network		
		mean	SD	max	mean	SD	max	mean	SD	max	mean	SD	max
Hong Kong	133	5.12	6.1	33	2.62	4.51	26	1.00	1.41	6	0.67	1.37	6
Indonesia	169	1.64	3.31	23	0.95	2.64	17	0.14	0.38	2	0.22	1.09	9
Japan	126	1.84	2.33	15	0.07	0.42	3	0.09	0.31	2	0.00	0.00	0
South Korea	133	2.5	2.8	21	1.09	1.37	6	0.15	0.40	2	0.02	0.15	1
Malaysia	281	7.35	6.61	37	1.07	1.94	8	2.15	3.09	18	0.36	0.74	5
Philippines	98	8.52	8.91	38	5.33	6.16	21	0.71	1.59	10	0.20	0.81	6
Singapore	116	3.52	3.24	15	0.59	1.66	12	1.28	2.40	11	0.57	1.90	14
Taiwan	107	1.6	2.22	12	0.21	1.11	7	0.14	0.46	3	0.00	0.00	0
Thailand	127	5.11	5.04	23	1.58	3.15	19	0.73	1.99	11	0.29	1.16	8

### Original Generated Statements

#### Entailed

- There are 9 different types country in the given table.
- The n value is same for Hong Kong and South Korea.
- There are 4 different types of Networks which contains its own mean, SD and max.
- The least max value is 0 in Political network of Taiwan.
- All the values of SD in Board network is greater than the values of SD in Family network.

#### Refuted

- All the values of SD in Board network is less than the values of SD in Family network.
- There are 4 different types of Networks which contains same mean, SD and max.
- The least max value is 0 in Political network of Thailand.
- There are 7 different types country in the given table.
- The n value is same for Hong Kong and Malaysia.

### Original Related Natural In-text Statements

- Descriptive statistics for each board network type are offered in Table 2, broken down by country.
- For each network interaction, there is considerable variation both across and within countries.

Figure 3: Sample crowd-sourced statements for one table (sourced from (Carney et al., 2020)). Please note that these are the original statements without any further corrections nor rephrasing.

grammatical errors as most BPO workers are not native English speakers. See Table 1 (first row) for detailed statistics of the crowd-sourced training set.

We initially attempted to collect the development and test sets as well as evidence annotations via the same method as the training set. However, we found that the quality was not gold-level and thus we (three of the authors) decided to manually correct the statements and annotate the evidence ourselves. All authors first annotated a small set of 102 statements to test inter-annotator agreement for statement relationship and evidence labeling. Out of 102 statements, we found 5 statements where at least one of three annotators disagreed on the relationship and a further 5 statements where the relationship was agreed but the evidence annotation differed. The other 92 were in complete agreement, indicating high agreement. Therefore, the annotations for the rest of the dev set were annotated by just one person. The test set was annotated fully by one author and the two other authors checked the annotations with all disagreements being resolved. See Figure 4 for a screenshot of the statement annotation correction and evidence annotation interface. See the third and fourth rows of Table 1 for detailed statistics of the dev and test sets.

### 3.3 Automatically generated statements

IBM Watson™ Discovery<sup>6</sup> is an AI-powered search and text analytics engine for extracting answers from complex business documents. One of the available functionalities is a Table Understanding service that produces a detailed enrichment of table data within an html document. We use this service to identify the body and header cells, as well as the *cell relationships*, within our dataset. We then proceed to use a set of templates to automatically create statements about each table. We begin by identifying which cells and columns are numeric and non-numeric using a simple regex. Unlike non-numeric cells, numeric cells and columns are appropriate for specific templates that expect numeric values, such as ‘Value [V] is the maximum of Column [C]’, where every value in column [C] has been identified as numeric. We also generate evidence for some of these templates. The template and evidence generation rules along with their inputs are detailed in Table 2. This process generated 3,512,978 statements from 1,980 tables which were highly skewed in favor of refuted statements. This dataset was then down-sampled to a maximum of 50 statements per table while ensuring a more even distribution between the two classes to form our final released auto-generated dataset. The

<sup>6</sup><https://www.ibm.com/cloud/watson-discovery>

Statement: "Los Aguanaces 3 other localities has same storage."

What is the statement relationship to the table? (required)

- Supported by cells in the table.
- Refuted by cells in the table
- Discard
- Unrelated to any cells in the table
- Need to discuss

Rephrase if needed

All Los Aguanaces localities have the same storage

Table 4

Studied material of Erinaceinae indet. and measurements. See for measuring details.

Locality	Code	MN	Local Zone	Age (Ma)	Sup./Inf.	Element type	Element nb.	Dex./Sin.	Storage	Catalogue nb.	Length (mm)	Width (mm)
Los Aguanaces 3	AG3	11	K	8.2	sup.	i	2	sin.	UU(MAP)	2102	1.73	1.13
Los Aguanaces 3	AG3	11	K	8.2	sup.	i	3	sin.	UU(MAP)	2103	2.22	1.72
Mas=a de la Roma 3	ROM3	9	I	10.1	sup.	m	2	dex.	UU(MAP)	308		
Los Aguanaces 3	AG3	11	K	8.2	sup.	m	3	dex.	UU(MAP)	2107	1.54	2.92
Patrimonio Forestal 5A	PF5A	11	J4	8.8	sup.	p	2	dex.	MAP	52		
Puente Minero 2	PM2	10	J2	9.7	sup.	p	4	sin.	UU(MAP)	201		

Statement: Los Aguanaces 3 other localities has same storage.

Select the cells in the table that support the relationship that you have determined for the above statement. Leave blank if you selected ambiguous or unrelated.

- There are 2+ different, conflicting sets of cells that relate to the statement

Can this table be used for evidence task B? (required)

- Yes
- No
- Need to discuss

Discussion:

Figure 4: Screenshot showing the labeling interface for statement rephrasing, relationship labeling and evidence annotation.

full statistics for the auto-generated training data is shown in the second row of Table 1.

## 4 Evaluation Metrics

### 4.1 Task A: Statement Fact Verification

The goal of task A is to determine if a statement is entailed or refuted by the given table, or whether, as is in some cases, this cannot be determined from the table. We show two evaluation results. The first is a standard 3-way Precision / Recall / F1 micro evaluation of a multi-class classification that evaluates whether each table was classified correctly as Entailed / Refuted / Unknown. This tests whether the classification algorithm understands cases where there is insufficient information to make a determination. The second, simpler evaluation, uses the same P/R/F1 metric but is a 2-way classification that removes statements with the “unknown” ground truth label from the evaluation. The 2-way metric still penalizes misclassifying refuted/ entailed statement as unknown.

### 4.2 Task B: Cell Evidence Selection

In Task B, the goal is to determine for each cell and each statement, if the cell is within the minimum set of cells needed to provide evidence for the statement (“relevant”) or not (“irrelevant”). In

other words, if the table were shown with all other cells blurred out, would this be enough for a human to reasonably determine that the table entails or refutes the statement? The evaluation calculates the recall and precision for each cell, with “relevant” cells as the positive category. For some statements, there may be multiple minimal sets of cells that can be used to determine statement entailment or refusal. In such cases, our dataset contains all of these versions. We compare the prediction to each ground truth version and count the highest score.

## 5 Experiments

We present our baseline experimental setup for each task below.

**Task A** We employ state-of-the-art Table-BERT implementation<sup>7</sup> as proposed by [Wenhu Chen and Wang \(2020\)](#). We utilize Table-BERT’s best performing configuration (Table-BERT-Horizontal-T+F-Template) as (1) using entity-linking to find the relevant columns for a statement, (2) flattening the table by scanning horizontally to form natural statements from the relevant columns and their cell values and (3) classifying the flattened table and the statement using the sentence pair classification

<sup>7</sup><https://github.com/wenhuchen/Table-Fact-Checking>

Input	Template	Evidence	Example Statements
col <sub>i</sub> , col <sub>j</sub>	'The' + col <sub>i</sub> _head + 'is' + col <sub>i</sub> _val + ', when the' + col <sub>j</sub> _head + 'is' + col <sub>j</sub> _val	col <sub>i</sub> _head, col <sub>j</sub> _head, col <sub>i</sub> _val, col <sub>j</sub> _val	The Code is AG3 when the Locality is Los Aguanances3.
col	col_val + 'is in' + col_head	col_val, col_head for en- tailed; col for refuted	AG3 is in Code.
col	unique or same values	col for entailed; None for refuted	Sup./Inf. has the same values.
col[#]	'The maximum of' + col_head + 'is'+val	col[#] for entailed; None for refuted	The maximum of Length(mm) is 2.22.
col[#]	'The minimum of' + col_head + 'is'+val	col[#] for entailed; None for refuted	The minimum of Length(mm) is 1.54.
col[#]	'The mean of' + col_head + 'is' + val	col[#]	The mean of Length(mm) is 1.83.
col[#]	'The median of' + col_head + 'is' + val	col[#]	The median of Length(mm) is 1.73.
col[#]	'The mode of' + col_head + 'is' + val	col[#]	The mode of Length(mm) is 1.54, 1.73, 2.22.

Table 2: Template and evidence rules used for auto-generated ground truth. The examples are derived from Table 4 in Figure 4.

setting in BERT. To overcome the lack of unknown statements in our dataset, we supplement each table with randomly chosen statements from other tables. In Table-BERT, if the entity linking results in no matches, the flattened table is marked as [UNK]. As our dataset contains unknown statements, in such cases we consider all columns to be a match and flatten the entire table.

Using the above process, we perform the following experiments (1) apply the Table-BERT model out-of-the-box (2) re-train Table-BERT model with unknown statement and apply on our test data (3) fine-tune the model in (2) with our manual+auto-generated data and apply on our test data. We also compare these experiments with a majority baseline with entailed as our majority class. The results are presented in Table 3. Applying Table-BERT model out-of-the-box provides some improvement over a majority-baseline. However, when the model is retrained with previously missing unknown statements, the performance improves for three-way classification. Further fine-tuning the model with our training dataset (both manual and auto-generated) provides the best performance on the two-way F1-score.

**Task B** We present the following two baselines for Task B: (1) a random baseline where each cell is marked relevant or irrelevant randomly (2) a simple word-match-based baseline where a cell is marked relevant if it overlaps with the statement. The baseline results are presented in Table 4.

Experiment	Test	
	2-way	3-way
majority-baseline	52.42	42.16
original Table-BERT	56.77	45.58
re-trained Table-BERT	52.96	48.33
+ FT with SEM-TAB-FACTS	56.81	48.24

Table 3: Task A baseline results using F1-score.

Experiment	Dev	Test
random-baseline	21.18	20.47
word-match	49.53	47.39

Table 4: Task B baseline results using F1-Score

## 6 Competition Results

We present two leaderboards for each task<sup>8</sup>. The official leaderboard is from participants who have given us detailed descriptions on their system and affirmed that they did not incorporate any information from the test set that changed their final model. This is a more accurate representation of system quality. The unverified leaderboard is composed of participants who either did not give enough detail or have affirmed that they incorporated some test data information in their final model. The participants did not have access to labels for test data but some teams altered their models upon examining

<sup>8</sup>We made the assumption that teams would not make any use of the test data, as is usually the case for algorithm evaluation, but we did not make this explicit ahead of time and some teams did not realize this was an issue. We decided to have two leaderboards to have a fair comparison for all teams.

Team	3-way F-Score	2-way F-Score
<b>Official Leaderboard</b>		
King001	<b>84.48</b>	<b>88.74</b>
THiFly_Queen	83.76	84.55
RyanStark	81.51	87.22
sattiy	77.32	84.96
BreakingBERT@IITK	69.31	76.81
Volta	67.34	72.89
TAPAS	66.81	73.13
AttesTable	65.59	71.72
Yaouxu	60.76	75.8
Beary-group	58.37	72.56
ok-team	57.79	71.84
SUNLP	47.92	59.58
FishToucher	41.83	52.01
KaushikAcharya	36.23	23.08
<b>Unverified Leaderboard</b>		
Skywalker	92.55	95.15
MagicPai	90.88	94.03
endworld	82.35	88.16
Paima	81.96	88.85
ravikranc	57.90	71.99

Table 5: Task A Leaderboard

the input data in the test set. Although we discouraged this approach, we present the results in hopes it can give some interesting information about how much improvement might be possible with having access to input test data.

19 teams participated in Task A. Of the 14 teams on the official leaderboard, King001 obtained the highest score for task A for both the 2-way (88.74) and 3-way (84.48) F-scores. However, the top three participants have comparable scores. All teams except for the last two beat our best baseline in Table 3. The unverified leaderboard includes 5 teams and contains higher scores than in the official leaderboard. However, due to the reasons outlined above, we cannot say with certainty that the results are reproducible. The full leaderboard results for all participants are in Table 5.

Task B is a much harder task and fewer teams participated in this challenge. Of the 12 teams that participated, 8 are in the official leaderboard. The best score is 65.17 by BreakingBERT@IITK(65.17) which is noticeably lower than the F-scores in Task A. Similarly to Task A the results in the unverified leaderboard are considerably higher. The full leaderboard results for all participants are in Table 6.

We summarize the system details for all participating teams in Tables 7 (Task A) and 8 (Task B). In general, deep learning was the most popular approach used by the participants e.g. BiL-

Team	F-Score
<b>Official Leaderboard</b>	
BreakingBERT@IITK	<b>65.17</b>
Volta	62.95
King001	62.14
FishToucher	60.06
RyanStark	54.96
Sattiy	48.56
AttesTable	43.02
KaushikAcharya	33.81
<b>Unverified Leaderboard</b>	
MagicPai	88.74
SkyWalker	73.05
endworld	57.85
Paima	51.97

Table 6: Task B Leaderboard

STM with attention, BERT (Devlin et al., 2019) etc. Most of the participants used transformer-based models to train their systems with flavors ranging from general-domain BERT (Devlin et al., 2019) to table-understanding specific versions like TAPAS (Herzig et al., 2020), TaBERT (Yin et al., 2020) and Table-BERT (Wenhu Chen and Wang, 2020). One third of the participants employed some form of ensembling technique in their submission.

Most of the participants have used the manually generated ground-truth in the development of their systems, with only one team not finding it useful. Further, a large percentage of participants have used the auto-generated ground truth in their systems with three teams not finding it helpful in their evaluation.

In terms of external resources, a majority of the participants used external table understanding resources in their systems. Further, most of the participants employed pre-processing techniques like acronym completion, removing special characters, etc... A substantial percentage of participants used techniques like incorporating word embeddings, entity resolution etc. Finally, a large number of participants used TabFact (Wenhu Chen and Wang, 2020) as an external dataset.

We also conducted additional analyses on participant submissions on the official leaderboard. We show through the average confusion matrix for Task A in Table 9 that the Unknown label was the most difficult. In fact, there were more unknown statements incorrectly labelled as entailed than were correctly categorized. Naturally, the statements with the lowest accuracy (< 25%) consist of mainly unknown statements, especially those statements

Team	Description
AttesTable (Varma et al., 2021)	Extended TAPAS to 3 classes by fine-tuning it. Employed a novel way of synthesizing “unknown” samples.
BreakingBERT@IITK (Jindal et al., 2021)	Ensemble models with TAPAS and TableBERT Transformers in a hierarchical two-step method for 3-way classification (unknown vs not unknown first)
Beary-group	Used TAPAS model with TabFact task, and added unique features. Employed preprocessing tricks like k-fold validation and replacing the characters and did hyperparameter tuning.
BOUN (Köksal et al., 2021)*	Used text augmentation techniques such as back translation and synonym swapping on the TAPAS model. Domain adaptation and joint learning using SemTabFacts and TabFact datasets.
endworld	Data Cleaning. Ensemble combining 80 instances of trained TaPas-Large and label smoothing.
FishToucher	Motivated by TaPas, used BERT and enriched the embedding layer with two new token type embeddings: row and column ids* (*The team mistakenly submitted an old model version, see paper for more accurate scores)
Kaushik Acharya (Acharya, 2021)	Parsed statements into candidate logical form; mapped result to handwritten rules, to then execute over relevant cells (identified using string matching and universal dependency parsing)
King001	Trained 20 instances of TaPas, SAT and Table-Bert for an ensemble of 60 models. Used preprocessing like acronym completion, rules to align the table content with the question content, label smoothing.
MagicPai	Multi-model training using models such as TaBERT, tapas.wikisql, tapas.TabFact, tapas_masklm. Finally rule amendments and aligning the distribution of training and test data
ok-team	TAPAS pretrained on TabFact with preprocessing of data (like transforming English numerals to Arabic numerals, removing special characters etc.)
Paima	Fine-tuned TAPAS optimized to perform window scanning on statement-related table data. Pre-processing to reduce abbreviations for table headers, and identifying operation expressions.
RyanStark	Multi-model TaBERT pretrained Model fusion. Pre-processing such as case and abbreviations.
Sattiy (Ruan et al., 2021)	Ensemble of 6 fine-tuned pre-trained models on the augmented data with content snap-shot input. Augmented the data provided by expanding the labels. Used Fast Gradient Method and added disturbance to the embedding layer to obtain a more stable word representation and a more general model.
SkyWalker	Deep learning, LPA rules, TAPAS dataset
SUNLP	BERT for sequence classification, transfer learning
TAPAS (Müller et al., 2021)	Ensemble of TAPAS (BERT-large-like) models: trained with a Mask-LM task on Wikipedia tables, intermediate pre-training data and TabFact data. Hierarchical two-step method for 3-way classification. Added neutral statements during training: random and by removing one of the evidence columns.
THiFly_Queen (Yuxuan et al., 2021)	Ensemble models in a hierarchical two-step method. 8-model to identify unknown statements and 9-model ensemble to classify entailed/refuted. Incorporated different ensemble weights for various statement types (count, superlative, unique).
Volta (Gautam et al., 2021)	Finetuned TAPAS that was pretrained on TabFact. Pre-processing to standardize multiple header rows to a single header.
Yaoxu	Added numeric and enumerate features to TAPAS and also statistic information (such as count) as a new row/column to the table.

Table 7: Descriptions of systems from participants for Task A. \*Note: Team BOUN did not participate in the official leaderboard.

Team	Description
BreakingBERT@IITK	An ensemble of an individual cell-based NLI approach and a similarity approach with the cells and statement
FishToucher	BERT CLS tokens for statement and table cells are used to determine cell relationships to each other, and the statement (for relevant cells)
Kaushik Acharya	Relevant cells are output as part of Task A
RyanStark	BOW approach with rules applied based on word matches in header and data cells.
Volta	Finetuned TAPAS for cell selection. Different models for entailed and refuted statements. Used transfer learning and header standardization.

Table 8: Descriptions of systems from participants for Task B (when provided)



	Refuted	Entailed	Unknown
Refuted	164	81	3
Entailed	46	226	2
Unknown	16	72	43

Table 9: Task A average confusion matrix

that have words overlapping with those in the table. Out of the entailed and refuted statements, ones that require numerical reasoning, like range, count or comparisons seemed to be most challenging. The statements with the highest accuracy ( $> 95\%$ ) generally had most words or numbers exactly overlapping with those in the table. In task B, out of the statements with less than 30% evidence F-score, 86% were ones with a refuted relationship. Conversely, the statements with greater than 70% F-score, 74% were ones with an entailed relationship. This shows that it is more difficult to find the most direct evidence to prove that a statement is refuted by a table than it is to show the positive evidence that a particular statement is supported by it. We believe this is an interesting line of research for future studies.

## 7 Conclusion and Future Works

In this paper, we presented the data and competition results for SEM-TAB-FACTS, Shared Task 9 of SemEval 2021. We created a large dataset via automated and crowdsourced fact verification as well as evidence finding for tables. Our 19 teams had a variety of techniques to tackle this unique but very relevant problem. The evidence finding scores are still quite low and have a large improvement potential. Additionally, the test set may be expanded in future versions of this task with a combination of manually generated, natural, and automated statements.

## References

Kaushik Acharya. 2021. Kaushikacharya at SemEval-2021 task 9: Candidate generation for fact verification over tables. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.

Ruobing Su Aylin Woodward and Shayanne Gal. 2020. What to know about the coronavirus outbreak in 17 charts and maps. *Business Insider*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large anno-](#)

[tated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Michael Cafarella, Alon Halevy, Hongrae Lee, Jayant Madhavan, Cong Yu, Daisy Zhe Wang, and Eugene Wu. 2018. Ten years of webtables. *Proceedings of the VLDB Endowment*, 11(12):2140–2149.

Richard W Carney, Travers Barclay Child, and Xiang Li. 2020. Board connections and crisis performance: Family, state, and political networks. *Journal of Corporate Finance*, 64:101630.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. [The PASCAL recognising textual entailment challenge](#). In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment, MLCW’05*, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Katherine East, Sara C Hitchman, Ioannis Bakolis, Sarah Williams, Hazel Cheeseman, Deborah Arnott, and Ann McNeill. 2018. The association between smoking and electronic cigarette use in a cohort of young people. *Journal of Adolescent Health*, 62(5):539–547.

Devansh Gautam, Kshitij Gupta, and Manish Shrivastava. 2021. Volta at SemEval-2021 task 9: Statement verification and evidence finding with tables using TAPAS and transfer learning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. [INFOTABS: Inference on tables as semi-structured data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324, Online. Association for Computational Linguistics.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

- Jane Hoffswell and Zhicheng Liu. 2019. Interactive repair of tables extracted from pdf documents on mobile devices. In *ACM Human Factors in Computing Systems (CHI)*.
- Aditya Jindal, Ankur Gupta, Jaya Srivastava, Preeti Menghwani, Vijit Malik, Vishesh Kaushik, and Ashutosh Modi. 2021. Breakingbert@iitk at SemEval-2021 task 9: Statement verification and evidence finding with tables. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI Conference on Artificial Intelligence*.
- Abdullatif Köksal, Yusuf Yüksel, Bekir Yıldırım, and Arzucan Özgür. 2021. BOUN at SemEval-2021 Task 9: Text Augmentation Techniques for Fact Verification in Tabular Data. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Tsvetomila Mihaylova, Georgi Karadzhov, Pepa Atanasova, Ramy Baly, Mitra Mohtarami, and Preslav Nakov. 2019. SemEval-2019 task 8: Fact checking in community question answering forums. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 860–869, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Thomas Müller, Julian Martin Eisenschlos, and Syrine Krichene. 2021. TAPAS at SemEval-2021 task 9: Reasoning over tables with intermediate pre-training. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*. International Committee for Computational Linguistics.
- Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Xiaoyi Ruan, mei Jin, Jian Ma, Lianxin Jiang, Mo Yang, and Jianping Shen. 2021. Sattiy at SemEval-2021 task 9: Method for statement verification and evidence finding with tables based on multi-model ensemble. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web, WWW '16*, page 771–782, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Harshit Varma, Aadish Jain, Pratik Ratadiya, and Abhishek Rathi. 2021. Attestable at SemEval-2021 task 9: Extending statement verification with tables for unknown class, and semantic evidence finding. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyu Zhou Wenhua Chen, Hongmin Wang and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.
- Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.
- Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics.
- Zhou Yuxuan, Zhou Kaiyin, Liu Xien, Wu Ji, and Zhu Xiaodan. 2021. Thify\_queen at SemEval-2021 task 9: Statement verification and evidence finding with tables. In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.

# BreakingBERT@IITK at SemEval-2021

## Task 9: Statement Verification and Evidence Finding with Tables

Aditya Jindal\*      Ankur Gupta\*      Jaya Srivastava\*  
Preeti Menghwani\*      Vijit Malik\*      Vishesh Kaushik\*  
Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)  
{adityaji, ankugupt, jayasri}@iitk.ac.in  
{priti, vijitvm, kvishesh}@iitk.ac.in  
ashutoshm@cse.iitk.ac.in

### Abstract

Recently, there has been an interest in factual verification and prediction over structured data like tables and graphs. To circumvent any false news incident, it is necessary to not only model and predict over structured data efficiently but also to explain those predictions. In this paper, as part of the SemEval-2021 Task 9, we tackle the problem of fact verification and evidence finding over tabular data. There are two subtasks. Given a table and a statement/fact, subtask A determines whether the statement is inferred from the tabular data, and subtask B determines which cells in the table provide evidence for the former subtask. We make a comparison of the baselines and state-of-the-art approaches over the given SemTabFact dataset. We also propose a novel approach CellBERT to solve evidence finding as a form of the Natural Language Inference task. We obtain a 3-way F1 score of 0.69 on subtask A and an F1 score of 0.65 on subtask B.

## 1 Introduction

Textual Inference, also known as natural language inference (Bowman et al., 2015), plays an important role in the study of natural language understanding and semantic representation. Due to the unprecedented amount of information generated over the internet, it becomes essential for machines to comprehend new information based on previous knowledge. Recent social events like political elections and pandemic spread have also shown the need for intelligent fact-checking systems that majorly depends on textual Inference over the scientific data.

Though Textual Inference is well explored, the current works mainly deal with unstructured Evidence in the form of sentences (Dagan et al., 2005).

\* Authors equally contributed to this work. Names in alphabetical order.

Verification under structured and semi-structured Evidence, such as tables, graphs, and databases, remains unexplored. Tables are ubiquitous in documents and presentations for concisely conveying important information; however, Inference on structured data like tables or graphs is much more difficult than simple text format due to complex structure and non-universal schema for the representation of data. Though recently, there has been work on Tabular Inference problems ( Zhong et al., 2020; Cho et al., 2018; Sun et al., 2018; Wenhua Chen and Wang, 2020; Eisenschlos et al., 2020; Pasupat and Liang, 2015; Wang et al., 2018 ) explaining the prediction, evidence finding is still an unexplored area.

Through the SemEval-2021 Task 9 (Wang et al., 2021) we have tried to solve the Tabular Inference problem over scientific tables by providing an answer as well as a solution to our reasoning. In other words, given the structured table data and statement, we aim to classify the statement as entailed, unknown (neutral), or contradiction. In addition, we also aim to classify each cell of the table whether it is relevant or irrelevant in making the aforementioned prediction. Our contribution is three-fold:

- We perform an empirical study of current state-of-the-art models on the SemTabFact dataset for the task of statement verification (see Section 5.1).
- We implement TableSciBERT, TableRoBERTa and develop a heuristic-based classifier. We achieve a 3-way F1 score of 0.69 on statement verification by ensembling TableSciBERT and TAPAS with our heuristic method (see Section 3.1).
- We propose a new model CellBERT, for the task of Evidence finding from the tables. We

achieve an F1 score of 0.65 on Evidence finding by ensembling CellBERT with our heuristic-based approach (See Section 3.2).

The code for all our experiments and pre-trained models are available on GitHub<sup>1</sup>.

## 2 Background

### 2.1 Related Work

Recently, [Wenhu Chen and Wang \(2020\)](#) proposed TabFact, a dataset with 16k Wikipedia tables and 118k human-annotated natural language statements, labeled as either ENTAILED or REFUTED. The authors proposed the TableBERT model for the task of fact-checking. TableBERT uses the pre-trained BERT ([Devlin et al., 2018](#)) model and fine-tunes it using the TabFact dataset as a simple NLI task by linearizing the table along with the fact. The linearized table is then concatenated with the statement which after tokenization is given as input to the BERT model which is used for binary classification to predict the nature of the statement.

The paper also proposed LPA (Latent Program Algorithm) to formulate the table fact-checking as a program synthesis problem. LPA uses reinforcement learning to optimize the task reward of this structured prediction problem directly, as was done in Neural Symbolic Machines (NSM) ([Liang et al., 2016](#)). [Zhong et al. \(2020\)](#) used the combination of the linguistic and symbolic reasoning integrated with an understanding of a given table’s structural format.

[Herzig et al. \(2020\)](#) developed the TAPAS model that performs question-answering over tables without generating logical forms. It uses weak supervision and predicts the answer by selecting table cells and optimally applying aggregation operators (for example: count, sum, average etc.) to the selected cells. An input instance to the model is the combination of the tokenized question and flattened table, separated by an [SEP] token (see Figure 1). In addition to BERT embeddings, TAPAS incorporates the table’s structural information via row, column, and rank embeddings. Since TAPAS uses flattened tables, it also suffers from the limitation of self-attention computation over long input sequences like BERT. Due to this reason, it fails to capture information over large tables or /databases containing/ multiple tables.

### Statement + Caption + Flattened Table

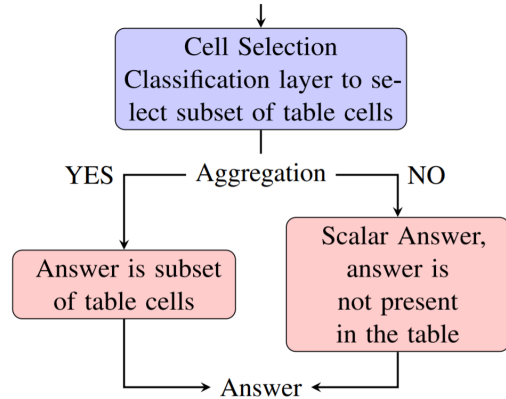


Figure 1: Flow chart explaining the functioning of TAPAS model using the two classification layers.

Besides this, the model’s expressivity is limited to a form of aggregation over a few cells of the table; hence, it fails to handle questions requiring multiple aggregation operations properly.

Recently, [Eisenschlos et al. \(2020\)](#) adapted TAPAS for the task of fact-checking. They introduced two intermediate pre-training tasks learned from the MASK-LM model. The first task is based on counterfactual statements, generated by creating one positive and one negative from every relevant Statement extracted from Wikipedia statements and tables. The second task is based on synthetic statements that generate a sentence by sampling from a set of logical expressions.

Let  $S$  and  $T$  represent the statement/fact and the table, respectively, which are given as input to the model. Furthermore, let  $E_S$  and  $E_T$  represent the corresponding input embeddings. The sequence of statement and the table given by  $E = [E_{[CLS]}; E_S; E_{[SEP]}; E_T]$  is passed through the transformer,  $f$  and a contextual representation is obtained for every token. The entailment probability  $P(S|T)$  is modeled using a single hidden layer neural network obtained by computing the output of [CLS] token:

$$P(S|T) = MLP(f_{[CLS]}(E)) \quad (1)$$

To handle large size tables, Table pruning is done using Heuristic exact match (HEM). In this method, the columns are ranked by a relevance score and added in order of decreasing relevance. Columns that exceed the maximum input length are skipped.

Our task for statement verification differs from the works mentioned above in the aspect that we have a third label, ‘unknown’, where the table fails

<sup>1</sup><https://github.com/vijit-m/TabEval>

to provide any information to infer the Statement. Moreover, no prior work has been performed concerning the task of evidence finding.

## 2.2 Problem Definition

The problem statement is articulated around the following two related subtasks.

**Subtask A - Table Statement Support:** Given a statement/fact, some of which will be directly adapted from the linking text, and a table, determine whether the table’s information supports the Statement. In this classification problem, a statement is assigned one of the following labels:

- **Fully Supported:** Statement is supported by data found within the table (denoted by 1).
- **Refuted:** Statement is contradicted by table (denoted by 0).
- **Unknown:** Not enough information in table to assess statement veracity (denoted by 2).

Mathematically, the problem can be described as, given a table  $\mathbb{T}$  and a statement  $\mathbb{S}$ , we need to learn a mapping  $\mathbb{F}_A$  to the output  $y_A$ , where  $y \in \{0, 1, 2\}$ . See examples in table 1 and table 2.

**Subtask B - Relevant Cell Selection:** Given a statement and a table, determine which table cells form relevant evidence for the Statement (if any). A table cell is evidence for a statement if it helps support or refute a part of the statement. In this subtask each cell of the table is assigned the following labels:

- **Relevant:** the cell must be included (denoted by 1).
- **Irrelevant:** the cell must not be included (denoted by 0).

Mathematically, each cell  $x_{ij} \in \mathbb{T}$  (where  $i, j$  correspond to row and column number respectively), needs to be assigned a value  $y_B \in \{0, 1\}$ . See examples in table 1 and table 2.

Body Sensation	Agoraphobic	Pleasant
number	museum	lovely
palpitation	shop	happiness
heartbeat	boat	Joyous

Table 1: <sup>2</sup> A sample table and statement with correct results for subtask B. **violet:** Relevant Cell, **red:** Irrelevant Cell

Statement	Label
Palpitation is a bodily sensation	Supported
Joyous and boat have same strength	Unknown
Lovely is an agoraphobic situation	Refuted

Table 2: Statements and Labels corresponding to Subtask A

**Corpus Collection:** The training and testing data is sourced from open-access scientific articles with tables using APIs provided by Science Direct for data mining. The data is procured in XML format and each table is also provided in image format since the size and styling of table contents are useful in understanding the table structure. Two separate datasets (with varying complexity) are provided, one in which the statements are automatically-generated, the second one where statements are generated manually by humans. The automatically generated statements are relatively more complex (Wang et al., 2021) and more in number as compared to the Manually generated statements.

**Annotation Process:** Each statement in the SemTabFact dataset is adapted from existing text and verified by at least one human reader. Multiple readers verify a smaller proportion to assess inter-annotator agreement.

**Data Preprocessing:** The Tables in the dataset have multiple sub-columns, unlike the Wikipedia tables in the TabFact dataset. The Training dataset had few errors like label classification errors and Grammatical errors. To overcome this, data is pre-processed and cleaned before feeding it into the models. During the Data Cleaning, statements with no labels are removed, and subcolumns are interpolated to handle the tables’ complex header structure.

The subcolumns are also merged with the table headers in the case of multiple Table headers that improve the results. The tables are provided with surrounding text along with the captions. During preprocessing, the surrounding text is combined with captions which together serve as captions to the subtask A model. Refer to Section 3.1 for our system description for subtask A. (See Appendix A for preprocessing details). We also discuss the result of this step on our model’s performance in Section 6

Source	#Tables	#Entailed	#Refuted	#Unknown	#Relevant	#Irrelevant
Manual	981	2818	1688	0	0	0
Auto-gen.	1980	92136	87209	0	1039058	15467957
Development	52	250	213	93	3048	28495
Test	52	274	248	131	3458	26724

Table 3: Dataset statistics for different datasets within SemTabFacts.

[CLS] + [Row\_Header] + [Cell Content] + [Column Header] + [SEP] + [Statement Text] + [SEP]

Figure 2: Approach for Subtask B

### 3 System Description

#### 3.1 Subtask A

Our proposed model for subtask A is an ensemble of TableSciBERT, TAPAS, and a heuristic-based approach. We first use a similarity-based approach to predict and segregate ‘unknown’ statements, and then we predict the remaining statements using an ensemble of TableSciBERT and TAPAS. Given a statement,  $S$  and table,  $T$ , in order to classify whether  $S$  is ‘unknown’ or not, we first calculate the similarity score between  $S$  and each cell  $C \subset T$  using equation 2. Here,  $Sim$  is a similarity function,  $C$  is the content of the cell,  $s_i$  is  $i$ -th token of the statement  $S$  after removing stop words and  $c_j$  is the  $j$ -th token of cell (for handling multi-word cells). We use the nltk library (Bird et al., 2009) to tokenize the statement and cell contents. The similarity function takes as input two tokens and outputs the similarity between them in 0 to 1 (higher score representing more similar). For each token in the statement, we first iterate through all the tokens of the cell and compute the maximum of scores obtained by  $Sim$  function for each token in the cell and the particular token of the statement. We then sum it over all the tokens of the statement to compute the score,  $s_c$ . We obtain the aggregated score,  $s_s$  over the whole table  $T$  by adding score  $s_c$ , of each cell  $c \subset T$  (see equation 3). Refer to Section 4.1 for more information about the types of similarity functions we experimented with.

$$s_c = \sum_i \max_j (Sim(c_j, s_i)), s_i \subset S, c_j \subset C \quad (2)$$

$$s_s = \sum_c s_c, \text{ where } C \subset T \quad (3)$$

We use  $s_s$  as the similarity score between statement  $S$  and table  $T$ . If  $s_s < \lambda_a$ , we label the

statement as ‘unknown’ where  $\lambda_a$  is a hyperparameter. If  $s_s \geq \lambda_a$  we proceed with the two-way classification using our ensemble of TableSciBERT and TAPAS.

Note that these TableSciBERT and TAPAS models were fine-tuned upon two labels only (viz-a-vis Entailed and Refuted). The ensemble is done by applying weighted average upon prediction probabilities of TAPAS and TableSciBERT. TableSciBERT (from TableBERT) was developed by replacing the BERT base model with SciBERT (Beltagy et al., 2019). SciBERT is a pre-trained language model based on BERT, which is fine-tuned upon large scale scientific data. Since the SemTabFact dataset is from scientific articles, using SciBERT makes sense intuitively.

TAPAS and TableSciBERT were trained on the training set (both autogenerated and manual dataset) and the development set. The table pruning method using the Heuristic exact match (HEM) was applied for the large complex tables in an Autogenerated dataset to handle the input embedding size. We also experimented with other transformers models like BioBERT (Lee et al., 2020), CovidBERT<sup>3</sup> but SciBERT gave the best result. Furthermore, we experimented with training them as a 3-label classifier as well. Refer to the Section 3.2 for details.

For training TableSciBERT and other table transformer 3-label variants we augmented the training data with statements having ‘unknown’ label, in order to balance the scarcity of unknown labels in the provided data. The augmentation for a table was done by randomly sampling statements from other tables provided in the dataset having ‘entailed’ or ‘refuted’ as the true label. The motivation behind using this strategy was that these statements served as statements with an ‘unknown’ label for this table. During sampling, we ensured that entailed and

<sup>3</sup><https://huggingface.co/gsarti/covidbert-nli>

refuted statements are equal in number to prevent any bias.

Overall, given a table  $\mathbb{T}$  and statement  $\mathbb{S}$ , the complete pipeline is a two-step process:

- We first perform the binary classification of whether  $\mathbb{S}$  is ‘unknown’ or not using the Similarity heuristic. If the predicted label is ‘unknown’, it is taken as the final prediction; otherwise, we proceed to the next step.
- We use the ensemble of TAPAS and TableSciBERT models to predict the ‘entailed’ or ‘refuted’ label of  $\mathbb{S}$ . Here,  $\mathbb{S}$  are the statements that were NOT classified as ‘unknown’ in the previous step.

### 3.2 Subtask B

For subtask B, we developed an ensemble of two different techniques:

- **CellBERT**: We propose a new method CellBERT as a BERT-base model that is fine-tuned upon an individual cell-based Natural Language Inference Task. We preprocess the training data given to us in Subtask B (which consists of only auto-generated statements) to generate NLI input samples of the form described in Figure 2. Mathematically, given a statement  $S$  and a table  $T$ , we need to label each cell  $c$  in the table as relevant or irrelevant. For CellBERT, each cell’s label is individually determined along with the supporting information of row and column headers. In other words, if the coordinates of a cell  $c$  are given by  $(x, y)$ , where  $x$  is the row number and  $y$  is the column number, the coordinates of the row header and column header cells are given by  $(x, 1)$  and  $(1, y)$  respectively.

The motivation behind using row and column headers information is that these capture the ‘type’ of data present in the cell. The combination of the row header, the cell, and the column header’s contents represent the NLI task’s premise. The hypothesis is taken as the statement provided. Note that using this approach, we ended up with around a million data points to train upon. Due to the unavailability of adequate computational resources, we restricted to using only 0.1% of the preprocessed training data.

- **Similarity**: We observed that Scientific Tables contained many entities for which pre-trained word embeddings are unavailable, and thus supervised approaches like CellBERT, fails to capture the required relationship. To overcome this, we used a cell-wise similarity algorithm, which calculates the score  $s_c$  of each cell  $C$  with the statement  $S$  same as in equation 2. We used  $s_c$  as the similarity score between statement  $S$  and cell  $C$ . If  $s_c < \lambda_b$  we label the cell as ‘irrelevant’, where  $\lambda_b$  is a hyperparameter. Otherwise, we label the cell as ‘relevant’.

## 4 Experiments

### 4.1 Subtask A

Following the TabFact’s TableBERT, we fine-tuned our own TableBERT model on the SemTabFact dataset for Subtask A. We also experimented with mutations of TableBERT by using RoBERTa (TableRoBERTa), XLNet (TableXLNet), and SciBERT (TableSciBERT) as well. We also experimented with implementing BiGRU layers on top of these table transformers. All our experiments were conducted using PyTorch (Paszke et al., 2019) Deep Learning library.

We experimented with a dataset ( $D_1$ ) which contained the Manual dataset along with the Auto-generated statements. A caveat to  $D_1$  was that auto-generated statements that have common tables with the manual dataset were only used. This was done because every model we trained upon only on the manual dataset was overfitting. The overfitting was due to an insufficient number of statements, i.e., 4056 in the Manual Dataset. After preparing the dataset  $D_1$ , we had a total of 72k statements.

For the similarity-based approach, we manually experimented with various non-semantic similarity approaches like edit distance and binary-matching<sup>4</sup> as well as embedding space-based semantic similarity approaches by first computing the word vector. We computed word vectors using GloVe (Pennington et al., 2014), BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). Cosine similarity and euclidean distance was used to compute similarity between two vectors. The non-semantic based binary-matching approach outperformed others upon the validation set; therefore, we used it

<sup>4</sup>We define binary matching score between two tokens  $t_1$  and  $t_2$  as 1 if the lower-cased, lemmatized and stemmed form of both the tokens is the same otherwise it is taken as 0.

to evaluate our results on the test set. In our final submitted model, SciBERT was fine-tuned for 3 epochs upon the combined dataset  $D_1$  and development dataset, with learning rates as  $5e - 5$ ,  $5e - 6$  and  $1e - 6$  for each epoch. Batch size was kept as 6 with maximum sequence length of 512 tokens. TAPAS was fine tuned upon the auto-generated, manual and development datasets separately for 6, 12 and 5 epochs respectively. Learning rate was kept the same as  $2e - 5$  for each epoch with maximum sequence length as 512. Dropout probability was set to 0.07. For ensemble, we used weights 0.7 and 0.3 for TAPAS and TableSciBERT respectively. We set the hyperparameter  $\lambda_a$  as 2 in our similarity heuristic.

## 4.2 Subtask B

For Subtask B, we preprocessed the dataset into input samples as shown in Figure 2, and fine-tuned a BERT base model. Since the number of statements given corresponding to the auto-generated dataset is large and also accounting for the fact that each cell in the table is a separate input example, the number of tuples of (cell, statement) were very large (over ten million data points). Therefore, only 0.1% of all tuples ( $\approx 30k$  data points) were used to train CellBERT, and the rest of the data was discarded. Note that the 0.1% of the data that we selected to train CellBERT was kept completely balanced with respect to true labels. We also experimented with including and not-including header information during fine-tuning as well. See table 5.

Here as well, for the similarity-based approach, we manually experimented with the same non-semantic similarity approaches like edit distance and binary-matching as well as embedding space-based semantic similarity approaches we used in Section 4.1. In Subtask B too, the non-semantic based binary-matching approach outperformed others upon the validation set, hence we used the same to evaluate our results on the test set. For our final model, the hyperparameter  $\lambda_b$  was set to 1. For CellBERT we fine-tuned a BERT base model for 5 epochs with batch size 16 and learning rate as  $2e - 5$ .

## 5 Results

### 5.1 Subtask A

The organizers use two evaluation metrics for subtask A:

- **3-way-F1:** This is a standard precision/recall evaluation (Three-Way) of a multi-class classification that evaluates whether each statement is classified as Entailed / Refuted / Unknown.
- **2-way-F1:** The second evaluation method is a Two Way method in which statements with the unknown ground truth label are not taken into consideration.

In both methods, first F1 scores are calculated for each table, which is then averaged across all tables for the final F1 score. The results of subtask A on the test set are shown in table 4:

Model	Task A 2-way F1	Task A 3-way F1
<b>(TAPAS+TableSciBERT)-2-Label+Similarity</b>	<b>0.7681</b>	<b>0.6931</b>
BERT-3-Label	0.5963	0.5295
TAPAS -2-Label + Similarity	0.7547	0.6824
SciBERT-2-Label + Similarity	0.6172	0.5534
RoBERTa-3-Label	0.6186	0.5271
(RoBERTa+BiGRU) 3-Label	0.5986	0.5113

Table 4: Test Data Result (Average F1-scores) for subtask A

Confusion Matrix of Test set on (TAPAS+TableSciBERT)-2-Label+Similarity is shown in fig. 3

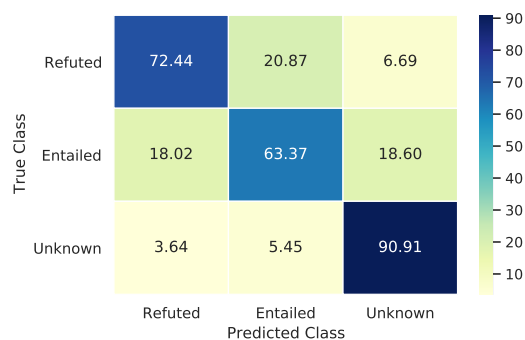


Figure 3: Confusion Matrix of Testset on (Tapas+TableSciBERT)-2-Label+Similarity.

The best results for subtask A were obtained using (TAPAS+TableSciBERT)-2-Label+ Heuristic-based Similarity with 0.768 as 2-way F1 and 0.693 as 3-way-F1. TAPAS-2-Label+ Heuristic Based Similarity also gave second-best results with 0.755 as 2-way F1 and 0.682 as 3-way-F1. We have experimented with other table transformers like



RoBERTa, BERT, and XLNet, but none of these gave promising results. Models like RoBERTa-3-Label, BERT-3-Label have been trained on the three labels and gave 0.527 and 0.529, respectively, as 3-way-F1. From the experiments, we observed that the models trained on 3 label classification performed poorly in classifying statements with ‘unknown’ label.

## 5.2 Subtask B

The metric used by the organizers calculates the recall and precision for each cell, with “relevant” cells as the positive category. Similar to Task A, the score is averaged over all statements in each table first, before calculating average across all tables. The results for subtask B on test data are shown in table 5. We obtained the best F1 of 0.6517. We have also shown the results of other variants of Cell-BERT, which are classified on the basis of row and column headers (see Table 5).

S.No	Method	F1
1.	Similarity	0.6414
2.	CellBERT	0.5380
3.	CellBERT - DevT	0.6465
<b>4.</b>	<b>CellBERT - DevT + Similarity Ensemble</b>	<b>0.6517</b>
5.	CellBERT (only cell context)	0.4891
6.	CellBERT (cell+row header information)	0.5213
7.	CellBERT (cell+column header information)	0.5199
8.	CellBERT (cell+row+column header information)	0.5380

Table 5: Test Data Result (Average F1-scores) for sub-task B

## 6 Error Analysis

**Subtask A:** On analyzing the training dataset, we realized that many statements require aggregation methods like sum, count, max, and min over the tabular data to determine whether a statement is entailed or refuted (or if it is ‘unknown’). It requires a symbolical understanding of the text that can not be understood using simple NLI based approaches

like Table-BERT and other table transformers. On the other hand, TAPAS outperforms other models primarily due to pre-training on the corpus of synthetic and counterfactual statements, as discussed in Section 2.1.

We noticed that TableSciBERT performed well as compared to other Table Transformers on sub-task A. It makes intuitive sense as the dataset is created from scientific texts and consequently has scientific statements. The organizers also mentioned that the training and testing data is sourced from open access scientific articles with tables using APIs provided by Science Direct for data mining in the task description.

Further, we improved the F1 score on the TAPAS model by 2% by using the multiple header merging technique (See Appendix A). The reason being that the merged headers have more semantic information as they contain the sub-headers too. We have explained the pre-processing step of tables with multiple headers and sub-columns in the Appendix. For 3-way classification, the table transformers gave unsatisfactory results, for example, RoBERTa 3-Label with F1 of 0.527 primarily due to two reasons, first we need complete domain knowledge of the related table to tag a statement to be unknown, and second, no data were available for the unknown label in the training set. We obtain 3-way F1 of 0.693, with (TAPAS+SciBERT)-2-Label+Similarity model, as our similarity heuristic was successfully able to classify 91% of the statements predicted unknown as true unknown labels. This might be possible because many unknown statements in the development and test set are completely independent of the table given.

**Subtask B:** We noticed that CellBERT performs best when header information is included while fine-tuning. An interesting point to note is that CellBERT trained only on autogenerated dataset provides an F1 score of 0.538, whereas when it was later fine-tuned upon the Development Dataset which had Manual based dataset (CellBERT-DevT), it boosted the results on the test set to F1 score of 0.646. The reason being the Test Dataset being a Manual based dataset, whereas the earlier CellBERT model was trained on the more complex autogenerated dataset.

The heuristic-based Similarity approach was performing surprisingly well on the test set as well,

giving us a comparable result of 0.641 F1. Although both the approaches are entirely different, the scores are comparable. This motivated us to analyze each method’s predictions, but we could not come up with a convincing hypothesis for what might be the reason for this observation. However, we noticed a common trend in both methods. When the number of cells in a table is large, and the number of relevant cells is very less, both models failed to identify relevant cells in such cases. In other words, both models had difficulty in identifying true positives.

## 7 Conclusion

This paper attempts a solution to an under-explored but essential problem: Statement Verification and Evidence Finding with Tables. There have been various works related to a binary classification of statements. Still, evidence finding for these classifications is a difficult and novel challenge. We are successfully able to present an ensemble of TAPAS model, table transformer-based TableSciBERT, and similarity heuristic trained for subtask A with 2-way F1 of 0.768 and 3-way F1 of 0.693. In subtask B, we implemented the CellBERT - DevT+ Similarity Ensemble method as our best model with an F1 score of 0.652.

In the future, we plan to progress in implementing new models that can tackle both linguistic and symbolic reasoning. We aim to extend the TAPAS model to 3 labels, requiring large data for training in unknown labels for good results. In the case of subtask B, we are planning to experiment with other NLI techniques and models. Besides, we will be looking into using more data for training CellBERT and other NLI models as well.

## References

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#).

Minseok Cho, Reinald Kim Amplayo, Seung won Hwang, and Jonghyuck Park. 2018. [Adversarial](#)

[TableQA: Attention supervision for question answering on tables](#).

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. [The PASCAL recognising textual entailment challenge](#). pages 177–190.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Julian Martin Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. [Understanding tables with intermediate pre-training](#).

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2016. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [PyTorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word](#)

- representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Yibo Sun, Duyu Tang, Nan Duan, Jingjing Xu, Xiaocheng Feng, and Bing Qin. 2018. Knowledge-aware conversational semantic parsing over web tables.
- Hao Wang, Xiaodong Zhang, Shuming Ma, Xu Sun, Houfeng Wang, and Mengxiang Wang. 2018. A neural question answering model based on semi-structured tables. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1941–1951, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 Task 9: Fact Verification and Evidence Finding for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyou Zhou Wenhui Chen, Hongmin Wang and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.
- Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. LogicalFactChecker: Leveraging logical operations for fact checking with Graph Module Network. *arXiv preprint arXiv:2004.13659*.

## Appendix

### A Preprocessing of the Multiple Header files

The provided dataset had many Tables with multiple headers and subcolumns, as shown in table 6.

Since most of our models take a single header as input only and with an equal number of columns in every row, we had to convert such tables to suit our input type. There were two processes involved for preprocessing such Tables.

**Intrapolation:** For one table, we would calculate the maximum numbers of columns in the Table and then interpolate all other rows with less columns to eventually have an equal number of columns in every row. Resulting table 7 obtained by interpolation of table 6 is shown below

ExperMatter	UserB		UserC	
	Base1	Base2	Base1	Base2
Gold	5.6		7	8
Silver	3.4	6.7	8.0	6.7

Table 6: Table with multiple headers and subcolumns

ExperMatter	UserB	UserB	UserC	UserC
	Base1	Base2	Base1	Base2
Gold	5.6		7	8
Silver	3.4	6.7	8.0	6.7

Table 7: Table after Intrapolation

**Header Merging:** Since the input our model has to be a single header file we had to merge such rows as shown below. Final table obtained after Preprocessing is shown in table 8.

ExperMatter	UserB	UserB	UserC	UserC
	Base1	Base2	Base1	Base2
Gold	5.6		7	8
Silver	3.4	6.7	8.0	6.7

Table 8: Table after preprocessing

### B Other Experiments with training data

We have also provided the results for 2-way F1 on the development set, which we have created out of the training dataset in table 9 were trained on only two labels. We divided both the Manual and Auto-generated training data into Train and Dev set with a split of 90 : 10, respectively. Various combinations of Train and Dev data were used to train different models. The models given in table 9 were trained on only two labels as both of the training datasets do not contain unknown labels, and this is also the reason why we have not shown 3-way F1.

TAPAS model gave us the best here too, with an F1 score of 97.1 when the Autogenerated train set and Dev set were used, while it gave us the best F1 score of 76.7 when the Autogenerated + Manual train set and Manual Dev set were used. We have not included any of the models trained only on the Manual train dataset because the provided Manual data is very small in numbers and the results of any model trained on manual train set were not very promising. Clearly TAPAS outperforms all the models and gives the best results on both the Datasets.

Model	Train set	Dev set	Metrics (On Dev set)			
			Precision	Recall	F1	Accuracy (%)
<b>TAPAS</b>	<b>Auto</b>	<b>Auto</b>	<b>99.4</b>	<b>94.7</b>	<b>97.1</b>	<b>94.32</b>
<b>TAPAS</b>	<b>Auto+Manual</b>	<b>Manual</b>	<b>82.6</b>	<b>71.5</b>	<b>76.7</b>	<b>70.83</b>
TableBERT	Auto	Auto	87.5	85.9	86.7	86.00
TableRoBERTa	Auto	Auto	63.5	63.1	63.3	64.05
TableRoBERTa + BiGRU	Auto	Auto	64.7	63.4	66.0	67.13
TableSciBERT	Auto	Auto	71.0	69.8	70.4	70.74
TableBERT	Auto	Manual	58.8	58.2	58.5	58.95
TableRoBERTa	Auto	Manual	52.9	50.7	51.8	51.95

Table 9: Results on the Development set created by us

# SemEval-2021 Task 12: Learning with Disagreements

Alexandra Uma<sup>1</sup> Tommaso Fornaciari<sup>2</sup> Anca Dumitrache<sup>3</sup> Tristan Miller<sup>4</sup>  
Jon Chamberlain<sup>5</sup> Barbara Plank<sup>6</sup> Edwin Simpson<sup>7</sup> Massimo Poesio<sup>1</sup>

<sup>1</sup>Queen Mary University of London <sup>2</sup>Università Bocconi, Milano <sup>3</sup>Albert Heijn

<sup>4</sup>Austrian Research Institute for Artificial Intelligence <sup>5</sup>University of Essex

<sup>6</sup>IT University of Copenhagen <sup>7</sup>University of Bristol

m.poesio@qmul.ac.uk

## Abstract

Disagreement between coders is ubiquitous in virtually all datasets annotated with human judgements in both natural language processing and computer vision. However, most supervised machine learning methods assume that a single preferred interpretation exists for each item, which is at best an idealization. The aim of the SemEval-2021 shared task on Learning with Disagreements (LE-WI-DI) was to provide a unified testing framework for methods for learning from data containing multiple and possibly contradictory annotations covering the best-known datasets containing information about disagreements for interpreting language and classifying images. In this paper we describe the shared task and its results.

## 1 Introduction

The assumption that natural language (NL) expressions have a single and clearly identifiable interpretation in a given context, or that images have a preferred labels, still underlies most work in NLP and computer vision. However, there is now plenty of evidence that this assumption is just a convenient idealization; virtually every project devoted to large-scale annotation has found that genuine disagreements are widespread.

In NLP, that annotator/coder disagreement can be genuine—i.e., resulting from debatable, difficult, or linguistic ambiguity—has long been known for anaphora and coreference (Poesio and Artstein, 2005; Versley, 2008; Recasens et al., 2011).<sup>1</sup> But in recent years, we have also seen evidence that disagreements among subjects/coders are common with virtually every aspect of language interpretation, from apparently simple aspects such as part-of-speech tagging (Plank et al., 2014b), to more

complex ones like semantic role assignment (Dumitrache et al., 2019), to subjective tasks such as sentiment analysis (Kenyon-Dean et al., 2018), and to the inferences that can be drawn from sentences (Pavlick and Kwiatkowski, 2019).

In computer vision, as well, the assumption that gold labels may be specified for items has proven an idealization (Rodrigues and Pereira, 2018)—in fact, possibly even more than for NLP. In many widely used crowdsourced datasets for computer vision, different coders assign equally plausible labels to the same items. The problem of disagreement among coders, including experts, on the classification of noisy image data has arisen in many CV applications. This includes classification of astronomical images (Smyth et al., 1994), medical image classification (Raykar et al., 2010), and numerous others (Sharmanska et al., 2016; Rodrigues and Pereira, 2018; Firman et al., 2018).

Many AI researchers have concluded that rather than attempting to eliminate disagreements from annotated corpora, we should preserve them—indeed, some researchers have argued that corpora should aim to collect all distinct interpretations of an expression (Smyth et al., 1994; Poesio and Artstein, 2005; Aroyo and Welty, 2015; Sharmanska et al., 2016; Plank, 2016; Kenyon-Dean et al., 2018; Firman et al., 2018; Pavlick and Kwiatkowski, 2019). Poesio and Artstein (2005) and Recasens et al. (2012) suggest that the best way to create resources capturing disagreements is by preserving *implicit* ambiguity—i.e., having multiple annotators label the items, and then keeping all these annotations, not just an aggregated ‘gold standard’. A number of corpora with these characteristics now exist (Pasonneau and Carpenter, 2014; Plank et al., 2014a; Dumitrache et al., 2019; Poesio et al., 2019; Rodrigues and Pereira, 2018; Peterson et al., 2019)

Much recent research has explored the question of whether corpora of this type, besides being more

<sup>1</sup>See also the analysis of disagreements in OntoNotes and word senses in Pradhan et al. (2012), Pasonneau et al. (2012), and Martínez Alonso et al. (2016).

accurate characterizations of the linguistic reality of language interpretation and image categorization, are also better resources for training NLP and computer vision models, and if so, what is the best way for exploiting disagreements in modeling. [Beigman Klebanov and Beigman \(2009\)](#) used information about disagreements to *exclude* items on which judgements are unclear (‘hard’ items). In the CrowdTruth project ([Aroyo and Welty, 2015](#); [Dumitrache et al., 2019](#)) information about disagreement is used to *weigh* the items used for training. [Plank et al. \(2014a\)](#) proposed to use the information about disagreement to *supplement* the gold label during training. Finally, methods were proposed for training directly from the data with disagreements, without first obtaining an aggregated label ([Sheng et al., 2008](#); [Rodrigues and Pereira, 2018](#); [Peterson et al., 2019](#); [Uma et al., 2020](#)). Only limited comparisons of these methods have been carried out ([Jamison and Gurevych, 2015](#)), and the sparse research landscape remains fragmented; in particular, methods applied in CV have not yet been tested in NLP, and vice versa.

The objective of SemEval-2021 Task 12, Learning with Disagreements (LE-WI-DI), was to provide a unified testing framework for learning from disagreements in NLP and CV using datasets containing information about disagreements for interpreting language and classifying images. The expectation being that unifying research on disagreement from different fields may lead to novel insights and impact AI widely.

## 2 Task organization

In order to provide a thorough benchmark for methods for learning from disagreements, we identified five well-known datasets for very different NLP and CV tasks, all characterized by providing a multiplicity of labels for each instance, by having a size sufficient to train state-of-the-art models, and by evincing different characteristics in terms of the crowd annotators and data collection procedure. We found or developed near-state-of-the-art models for the tasks represented by these datasets. Both ‘hard’ and ‘soft’ evaluation metrics were employed ([Uma et al., n.d.](#)).

The shared task was set up on the CodaLab Competitions platform,<sup>2</sup> which enables training and uniform evaluation on these datasets, such

<sup>2</sup><https://www.microsoft.com/en-us/research/project/codalab/>

that the crowd learning adaptations of the base models proposed by participants to the task would be directly comparable.

In this section, we briefly introduce the five datasets included in the benchmark and our evaluation criteria. We also elaborate on the setup of the shared task.

### 2.1 Data

There are by now quite a few datasets preserving disagreements, and covering many levels of language interpretation; remarkably, none of these has ever been used for a shared task like the one we are proposing, and the majority of them have never been used for a shared task at all. Our shared task has aimed at leveraging this diversity. The datasets included are outlined in this section and their characteristics are summarized in Table 1. Figure 1 shows the observed agreement of each dataset.

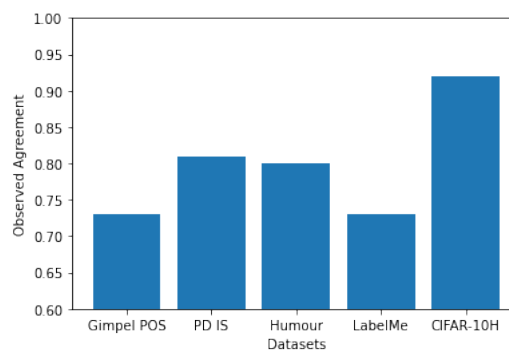


Figure 1: Observed Agreement for each dataset

#### 2.1.1 The Gimpel et al. POS corpus

One widely used resource for developing disagreement-aware NLP models is the dataset of Twitter posts annotated with POS tags collected by [Gimpel et al. \(2011\)](#). [Plank et al. \(2014b\)](#) mapped the Gimpel tags to the universal POS tag set ([Petrov et al., 2012](#)) and collected at least five crowdsourced labels per token from 177 annotators. This dataset contains 14K training examples (English words/tokens) annotated by 177 annotators. Each item was annotated between five and 177 times, 16.38 times on average. For this shared task, we selected 8.3K, 3K, and 3.1K tokens as training, development and test sets respectively.

#### 2.1.2 The pDIS corpus

The *Phrase Detectives* corpus ([Poesio et al., 2019](#)) is a crowdsourced coreference corpus collected

	POS	PDIS	HUMOUR	IC-LABELME	CIFAR-10H
Number of items	14,000	96,305	18,002	10,000	10,000
Number of crowd workers	177	1,741	272	59	2,457
Number of categories	12	2	2	8	10
Average annotations per item	16.37	11.87	5.00	2.50	51.10

Table 1: Summary of dataset characteristics

with the *Phrase Detectives* gamified online platform (Poesio et al., 2013).<sup>3</sup> We use PDIS, a simplified version of the corpus containing only binary information status labels: Discourse New (the entity referred to has never been mentioned before) and Discourse Old (it has been mentioned). PDIS consists of 542 documents, for a total of 408K tokens and over 96K markables. These documents were annotated by game players who produced an average of 11.87 annotations per markable.

Forty-five of the documents (5.2K markables), collectively called PD<sub>gold</sub>, additionally contain expert-adjudicated gold labels. This subset of PDIS was designated as the test set. The training and development datasets consist of 473 documents (and 86.9K markables) and 24 documents (4.2K markables) respectively.

### 2.1.3 The Humour dataset

The comprehension and appreciation of humour is known to vary across individuals (Ruch, 2008), making disagreement over the perceived funniness of jokes an appealing subject of study. For our training data, we used the corpus of Simpson et al. (2019), which consists of 4,030 short texts (3398 jokes, mostly based on puns, and 632 non-jokes such as proverbs and aphorisms). 28,210 unique pairings of these texts were presented to five crowdsourcers each, who indicated which text in the pair (if either) they found to be funnier. The goal is to learn a model that can predict binary pairwise labels that can predict which of two short texts is funnier.

The 4,030 text instances were split into 60% (2,418 texts, 9,916 unique pairs) for the training set and 20% (806 texts, 1,086 unique pairs) for the development set. Since this dataset has already been published, we constructed a new test dataset along similar lines: 1,000 short texts (all punning jokes) were paired in 7,000 different ways, and each of these 7,000 pairs was then presented to five crowd workers for a preference judgement.<sup>4</sup>

<sup>3</sup><https://github.com/dali-ambiguity>

<sup>4</sup>US-based workers from Amazon Mechanical Turk were

### 2.1.4 The LabelMe corpus

Much research on learning from disagreements was motivated by computer vision datasets, so we intended to include some of these, too. Possibly the most widely used such corpus is the LabelMe dataset<sup>5</sup> (Russell et al., 2008). It classifies outdoor images according to 8 categories: *highway, inside city, tall building, street, forest, coast, mountain or open country*. Using Amazon Mechanical Turk, Rodrigues and Pereira (2018) collected an average of 2.5 annotations per image from 59 annotators for 10K images in this dataset.

We randomly selected 5K, 2.5K, and 2.5K images for training, development, and testing respectively, careful to keep the label proportions in each subset close to the proportions in the 10K dataset.

### 2.1.5 The CIFAR-10H corpus

Krizhevsky’s (2009) CIFAR-10 dataset consists of 60K tiny images from the web, carefully labelled and expert-adjudicated to produce a single gold label for each image in one of 10 categories: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck*. Peterson et al. (2019) collected crowd annotations for 10K images from this dataset (the designated test portion) using Amazon Mechanical Turk, creating the CIFAR-10H dataset<sup>6</sup> which we use for this shared task.

We randomly selected 7K, 1K, and 2K images for training, development and testing respectively. We kept as much data as we could for training without jeopardizing the evaluation process, as the base model was found to be sensitive to data size. As with the original dataset, each subset we created contains an equal number of images per category.

## 2.2 Evaluation metrics

While recent research questions the assumption that a single ‘hard’ label (a gold label) exists for every

employed, paid in line with the federal minimum wage.

<sup>5</sup><http://labelme.csail.mit.edu/Release3.0>

<sup>6</sup><https://github.com/jcpeterson/cifar-10h>



item in a dataset, the models proposed for learning from multiple interpretations are still largely evaluated under this assumption, using ‘hard’ measures like accuracy or class-weighted  $F_1$  (Sheng et al., 2008; Plank et al., 2014a; Martínez Alonso et al., 2015; Sharmanska et al., 2016; Rodrigues and Pereira, 2018). For reference and comparison reasons, we also evaluate the models produced for this shared task using  $F_1$ .

However, a way of evaluating models as to their ability to capture disagreement is needed, especially for datasets with substantial extent of disagreement. The simplest ‘soft’ metric of this type is to evaluate ambiguity-aware models by treating the probability distribution of labels they produce as a **soft label**, and comparing that to the full distribution produced by annotators, using, for example, cross-entropy. This approach was adopted in, *inter alia*, (Peterson et al., 2019; Uma et al., 2020). Peterson et al. (2019) tested this approach on image classification tasks, generating the soft label by transforming the item annotation distribution using standard normalization. In this shared task we also use standard normalization to produce soft labels for the humour dataset. Uma et al. (2020) show that the choice of soft label encoding function depends on the characteristics of the dataset. For POS and IC-LABELME, they show that a softmax function over the annotator distribution is preferable over standard normalization. On the other end, for PDIS, training a soft-loss model using the posterior probability produced by Hovy et al.’s (2013) MACE probabilistic aggregation model as a soft label produces predictions that are most accurate with respect to the gold.

Therefore, in this shared task we used different soft label encoders to generate soft labels from annotator distributions for the test data.

### 2.3 Task setup

CodaLab was the designated site for hosting SemEval-2021 competitions.<sup>7</sup> LE-WI-DI was run in two main phases:

**Practice phase.** In the practice phase, the goal was to train models for each task to learn from crowd annotations, given (1) the training data (consisting of raw and preprocessed input data and crowd annotations), (2) the development data with no labels, and (3) the base models (discussed in Section 3). While participants were encouraged to start with the

<sup>7</sup>Our competition can be found at <https://competitions.codalab.org/competitions/25748>.

base models and extend them, we did not make this mandatory. Participants could test the performance of their models on the development set by making predictions on the given development input data and then uploading their submissions to CodaLab for preliminary testing. We permitted up to 999 submissions in this phase. The ‘leader board’ was made public to allow participants not only to see how their models performed, but also to compare the performance of their model to those submitted by other participants.

**Evaluation phase.** The evaluation phase was the official testing phase of the competition. In this phase, we released test data (without labels) but we also released the gold labels and crowd annotations for the development set to facilitate quick offline testing and refining of models and model selection. The number of submissions for this phase was limited to ten submissions per participant to prevent the participants from fine-tuning their models on the test data.<sup>8</sup> The allowed number of submissions was later increased to 999 to more encourage submission attempts. The leader board was also kept public in this phase. Each participant could see the best model of each of the tasks using each of the evaluation metrics.

**Post-campaign evaluation.** As our aim was to make this benchmark available beyond the competition to researchers developing disagreement-aware models, we included a third, post-evaluation phase to allow lifetime access to the data. Researchers participating in this phase will be able to access the same data as in the evaluation phase and test their models on the test data for the various tasks.

## 3 Base models and baselines

In order to encourage the participants to focus on the development of methods for learning from disagreement, as opposed to achieving higher performance by developing better models, we provided ‘base’ models for each of the tasks represented by the aforementioned corpora. In this section, we briefly discuss the baseline models for each task that we provided. In Section 5, we report the results using these base models and two crowd learning approaches: majority voting and the soft loss method (Peterson et al., 2019; Uma et al., 2020).

<sup>8</sup>This proved unnecessary as the inherent difficulty of the shared task was enough of a deterrent.

**The pos tagging model.** The POS tagger is a bi-LSTM (Plank et al., 2016) with additional use of attention over the input word and character embeddings, as used in Uma et al. (2020).

**The pDIS classification model.** The model for this task was developed by comparing architectures from two models: a state-of-the-art coreference model and a state-of-the-art IS classification model. We combined the mention representation component of Lee et al.’s (2018) coreference resolution system with the mention sorting and non-syntactic feature extraction components of the IS classification model proposed by Hou (2016)<sup>9</sup> to create a novel IS classification model that outperforms Hou (2016) on the pDIS corpus. The training parameters were set following Lee et al. (2018).

**The humour preference learning model.** We use as base model for this task Gaussian process preference learning (GPPL) with stochastic variational inference, as described and implemented by Simpson and Gurevych (2020). As an input vector to GPPL, we first take the mean word embedding of a text, using 300-dimensional word2vec embeddings trained on the Google News corpus (Mikolov et al., 2013). Then, we compute the frequency of each unigram in the text in a 2017 Wikipedia dump, and each bigram in the text in a Google Books Ngram dataset. Finally, we concatenate the mean unigram and bigram frequencies with the mean word embedding vector to obtain the input vector representation for each short text. The GPPL model is trained on pairwise labels from the training set to obtain a ranking function that can be used to score test instances or output pairwise label probabilities. As a Bayesian model, it takes into account sparsity and noise in the crowdsourced training labels, and moderates its confidence accordingly. Hence, it is a strong baseline for accounting for disagreement among annotators. This same GPPL approach set the previous state of the art on the humour dataset (Simpson et al., 2019).

**The LabelMe image classification model.** For this task, we replicated the model from Rodrigues and Pereira (2018). The images were encoded using pretrained CNN layers of the vgg-16 deep neural network (Simonyan et al., 2013). This encoding is passed into a feed-forward neural network layer

<sup>9</sup>This model was developed for fine-grained information status classification on the ISNOTES corpus (Markert et al., 2012; Hou et al., 2013).

with a ReLU activated hidden layer with 128 units. A 0.2 dropout is applied to this learned representation which is then passed through a final layer with softmax activation to produce the model’s predictions.

**The CIFAR-10 image classification model.** The trained model provided for this task is the ResNet-34A model (He et al., 2016), a deep residual framework which is one of the best performing systems for the CIFAR-10 image classification. We made available to participants the publicly available Pytorch implementation of this ResNet model.<sup>10</sup>

## 4 Participating systems

Unfortunately, we observed a dramatic difference in the number of participants that signed up to the competition (over 100 groups), the number of groups that participated in the trial phase, and the number of groups that submitted a run for official evaluation.<sup>11</sup> Only one group, UOR, submitted in the evaluation phase (Osei-Brefo et al., 2021). However, they did submit models for each of the tasks, and did adopt a learning from disagreements approach.

**pos tagging.** For POS tagging, UOR developed a novel POS tagging model by fine-tuning the BERT language model (Devlin et al., 2019). The (tweet, token) pairs were encoded in the form

[CLS] Tweeted text [SEP] Token [SEP]

where the ‘[CLS]’ token was added for classification and the ‘[SEP]’ token separated the tweet from the token under consideration. To learn the class for the token, the learned classification token was passed through a single feed-forward neural network layer with softmax activation. The output of this layer represented the probabilities of the token belonging to each of the 12 classes.

To extend this model for crowd learning, UOR added an adaptation of the crowd layer from Rodrigues and Pereira (2018). Rather than compute a single loss from the crowd layer as Rodrigues and Pereira (2018) do, UOR compute a joint loss from both the crowd layer and the base model (without the crowd layer bottleneck).

<sup>10</sup><https://github.com/KellerJordan/ResNet-PyTorch-CIFAR10>

<sup>11</sup>Two participating groups cited an inability to come up with a novel crowd learning paradigm as the reason they did not submit for official evaluation.

**PDIS classification.** For this task, UOR also used a fine-tuned BERT together with [Rodrigues and Pereira’s \(2018\)](#) crowd layer. Each (document, markable) pair was encoded as follows:

[CLS] + Document + [SEP] + Markable + [SEP]

where the ‘[CLS]’ and ‘[SEP]’ tokens are used in the same manner as in POS tagging.

**Humour preference learning.** For humour preference learning, the participant submitted predictions using the base model without modifications.

**LabelMe image classification (IC-LABELME).** For this task, UOR adapted the [Rodrigues and Pereira \(2018\)](#) crowd layer to the base model.

**CIFAR-10H image classification (IC-CIFAR10H).** For IC-CIFAR10H, the crowd labels were aggregated into hard labels using majority voting. However, UOR combined [Zagoruyko and Komodakis’s \(2016\)](#) WideResNet model, which has been shown to outperform [He et al.’s \(2016\)](#) ResNet with the novel Sharpness-Aware Minimization (SAM) optimization technique, proposed by [Foret et al. \(2020\)](#), that has been shown to efficiently improve model generalization, especially on noisy, singly labelled data.

## 5 Results and discussion

Table 2 contains the results of various models discussed in Sections 3 and 4 on this shared task when evaluated based on the hard metric (i.e., the class-weighted  $F_1$  with respect to the gold labels) and the soft metric (the cross-entropy between the soft labels for each task—see Section 2.2—and the model prediction for that task). The best results for each task are highlighted in bold.

UOR concentrated their effort on the IC-CIFAR10H dataset, on which they did achieve good results and outperformed the baseline (see below). In the other datasets, their official results at the end of the evaluation phase were less competitive.

With the POS and PDIS datasets, the model proposed by UOR, adding a crowd layer on top of BERT, achieved substantially worse results than training from a label aggregated using majority voting or training using a soft-loss function, both according to the hard evaluation metric ( $F_1$ ) and the soft metric (CE). The ranking between soft-loss method, aggregation, and crowd layer with POS is consistent with that obtained by [Uma et al. \(n.d.\)](#), but the results obtained by UOR are much worse for reasons that will require further investigation. (With PDIS,

[Uma et al. \(n.d.\)](#) obtain comparable results with soft-loss functions and with the crowd layer.) More generally, the results show that although the hard label (the majority voting aggregate of the annotator distribution) and the soft label (a probability distribution encoding of the annotator distribution) were drawn from the same annotator distribution with this dataset, given the same base model, training by targeting the soft label (base model + soft loss) outperforms training using majority voting aggregates (base model + majority voting) regardless of which evaluation metric is used to compare the models.

For the humour preference learning task, again, the base model outperforms UOR’s submission on both metrics, but in this case the difference in performance between GPPL and UOR is much less substantial with the hard metric, although it remains large according to the soft metric. This large difference may be due to a technical issue that requires further investigation, since UOR’s submission was also supposed to have been produced by the same base system. A possible reason for poor cross-entropy error is the use of discrete labels, which are heavily penalized for overconfidence by cross-entropy error. On this soft metric, the Bayesian probabilistic approach of GPPL may have advantages over approaches with poorer calibration, which remains to be explored in future work. The GPPL approach therefore remains the state of the art with this dataset.

For IC-LABELME, again, soft-loss training achieved better hard and soft scores than both aggregation training with majority voting labels and the UOR extension of the base model using a crowd layer adapted from [Rodrigues and Pereira \(2018\)](#). The finding that the UOR group’s adaption of the [Rodrigues and Pereira \(2018\)](#) crowd layer yielded lower  $F_1$  than training using majority voting is unexpected, given that in [Rodrigues and Pereira \(2018\)](#); [Uma et al. \(2020\)](#) and [Uma et al. \(n.d.\)](#), the crowd layer, particularly the DL-MW variant, was shown to be a competitive approach to learning from crowds and always outperforms majority voting. However, UOR’s crowd layer does achieve better soft evaluation (cross-entropy) scores than majority voting.

There is one dataset, however, on which UOR outperformed the two baselines: IC-CIFAR10H. For this dataset, UOR used [Zagoruyko and Komodakis’s \(2016\)](#) WideResNet image classifier trained using majority voting aggregated labels and

Task	Model	Hard score (F <sub>1</sub> )	Soft score (cross-entropy)
POS	base model + majority voting	0.753	2.263
POS	base model + soft loss	<b>0.767</b>	<b>1.084</b>
POS	UOR (BERT + Crowd Layer)	0.125	2.331
PDIS	base model + majority voting	0.906	0.397
PDIS	base model + soft loss	<b>0.928</b>	<b>0.273</b>
PDIS	UOR (BERT + Crowd Layer)	0.474	0.830
HUMOUR	base model (GPPL)	<b>0.557</b>	<b>0.728</b>
HUMOUR	UOR	0.513	3.697
IC-LABELME	base model + majority voting	0.806	2.833
IC-LABELME	base model + soft loss	<b>0.833</b>	<b>1.691</b>
IC-LABELME	UOR (base model + Crowd Layer)	0.784	1.769
IC-CIFAR10H	base model + majority voting	0.646	2.610
IC-CIFAR10H	base model + soft loss	0.698	1.052
IC-CIFAR10H	UOR (WideResNet + SAM)	<b>0.769</b>	<b>0.827</b>

Table 2: Results on the benchmarks and participant submissions on all the tasks using F<sub>1</sub> (higher is better) and cross-entropy (lower is better)

Foret et al.’s (2020) SAM optimization technique. The results show that WideResNet outperforms ResNet with this task both according to the hard metric and the soft metric. Interestingly, this is the one dataset in which the Deep Learning from Crowds approach of Rodrigues and Pereira (2018) works best according to Uma et al. (n.d.), outperforming both soft-loss training and majority voting training. It would thus be interesting to understand if the performance of UOR’s model could be further increased by adopting one of these methods.<sup>12</sup>

## 6 Conclusion

This shared task presented the first unified testing framework for learning with disagreements. The datasets include sequence labelling, three classification tasks, and preference learning, hence provide a testbed for a wide range of challenges when learning from multiple annotators. We proposed to evaluate not just the ‘hard’ performance against a gold standard, but also the ability to predict the distribution of different interpretations of the data—that is, the alternative labellings provided by different annotators. The results show the benefit of soft loss functions that account for the distribution of labels in the training data. However, modelling alternative

<sup>12</sup>As a postscript, we should note that after the end of the official competition we did carry out an investigation of the reasons for the poor performance of UOR’s models on the tasks other than IC-CIFAR10H. Some points emerging from the discussion are presented in the participants’ paper for the shared task.

interpretations of data remains an under-researched topic in NLP and computer vision. To encourage future work on learning with disagreements, the shared task and datasets will remain available for evaluating new methods.

## Acknowledgments

Alexandra Uma, Jon Chamberlain, and Massimo Poesio were partially supported by the DALI project, ERC Advanced Grant 695662. Tristan Miller was supported by the Austrian Science Fund (FWF) under project M 2625-N31. Barbara Plank is supported in part by the Independent Research Fund Denmark (DFF) grant 9131-00019B and 9063-00077B.

## References

- Lora Aroyo and Chris Welty. 2015. *Truth is a lie: Crowd truth and the seven myths of human annotation*. *AI Magazine*, 36(1):15–24.
- Beata Beigman Klebanov and Eyal Beigman. 2009. *From annotator agreement to noise models*. *Computational Linguistics*, 35(4):495–503.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, pages 4171–4186. Association for Computational Linguistics.
- Anca Dumitrache, Lora Aroyo, and Chris Welty. 2019. *A crowdsourced frame disambiguation corpus with*

- ambiguity. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, pages 2164–2170. Association for Computational Linguistics.
- Michael Firman, Neill D. F. Campbell, Lourdes Agapito, and Gabriel J. Brostow. 2018. **DiverseNet: When one right answer is not enough**. *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5598–5607.
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. **Sharpness-aware minimization for efficiently improving generalization**. *CoRR*, abs/2010.01412.
- Kevin Gimpel, Nathan Schneider, Brendan O’Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. **Part-of-speech tagging for Twitter: Annotation, features, and experiments**. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 42–47. Association for Computational Linguistics.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. **Deep residual learning for image recognition**. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.
- Yufang Hou. 2016. **Incremental fine-grained information status classification using attention-based LSTMs**. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1880–1890. The COLING 2016 Organizing Committee.
- Yufang Hou, Katja Markert, and Michael Strube. 2013. **Global inference for bridging anaphora resolution**. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 907–917. Association for Computational Linguistics.
- Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. **Learning whom to trust with MACE**. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1120–1130. Association for Computational Linguistics.
- Emily Jamison and Iryna Gurevych. 2015. **Noise or additional information? Leveraging crowdsourcing annotation item agreement for natural language tasks**. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 291–297. Association for Computational Linguistics.
- Kian Kenyon-Dean, Eisha Ahmed, Scott Fujimoto, Jeremy Georges-Filteau, Christopher Glasz, Barleen Kaur, Auguste Lalande, Shruti Bhandari, Robert Belfer, Nirmal Kanagasabai, Roman Sarrazingendron, Rohit Verma, and Derek Ruths. 2018. **Sentiment analysis: It’s complicated!** In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 1, pages 1886–1895. Association for Computational Linguistics.
- Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>.
- Kenton Lee, Luheng He, and Luke Zettlemoyer. 2018. **Higher-order coreference resolution with coarse-to-fine inference**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 687–692. Association for Computational Linguistics.
- Katja Markert, Yufang Hou, and Michael Strube. 2012. **Collective classification for fine-grained information status**. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 795–804. Association for Computational Linguistics.
- Héctor Martínez Alonso, Anders Johannsen, and Barbara Plank. 2016. **Supersense tagging with inter-annotator disagreement**. In *Proceedings of the 10th Linguistic Annotation Workshop*, pages 43–48. Association for Computational Linguistics.
- Héctor Martínez Alonso, Barbara Plank, Arne Skjærholt, and Anders Søgaard. 2015. **Learning to parse with IAA-weighted loss**. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1357–1361. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. **Distributed representations of words and phrases and their compositionality**. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2, pages 3111–3119. Curran Associates Inc.
- Emmanuel Osei-Brefo, Thanet Markchom, and Huizhi Liang. 2021. UOR at SemEval-2021 Task 12: On crowd annotations; learning with disagreements to optimise crowd truth. In *Proceedings of the 15th International Workshop on Semantic Evaluation*. To appear.
- Rebecca J. Passonneau, Vikas Bhardwaj, Ansa Sallab-Aouissi, and Nancy Ide. 2012. **Multiplicity and word sense: evaluating and learning from multiply labeled word sense annotations**. *Language Resources and Evaluation*, 46(2):219–252.
- Rebecca J. Passonneau and Bob Carpenter. 2014. **The benefits of a model of annotation**. *Transactions of the Association for Computational Linguistics*, 2:311–326.
- Ellie Pavlick and Tom Kwiatkowski. 2019. **Inherent disagreements in human textual inferences**. *Transactions of the Association for Computational Linguistics*, 7:677–694.

- Joshua C. Peterson, Ruairidh M. Battleday, Thomas L. Griffiths, and Olga Russakovsky. 2019. [Human uncertainty makes classification more robust](#). In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision*, pages 9616–9625.
- Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. [A universal part-of-speech tagset](#). In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2089–2096. European Language Resources Association.
- Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. In *Proceedings of the 13th Conference on Natural Language Processing (KONVENS 2016)*.
- Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014a. [Learning part-of-speech taggers with inter-annotator agreement loss](#). In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 742–751. Association for Computational Linguistics.
- Barbara Plank, Dirk Hovy, and Anders Søgaard. 2014b. [Linguistically debatable or just plain wrong?](#) In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 507–511. Association for Computational Linguistics.
- Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. [Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 412–418. Association for Computational Linguistics.
- Massimo Poesio and Ron Artstein. 2005. [The reliability of anaphoric annotation, reconsidered: Taking ambiguity into account](#). In *Proceedings of the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky*, pages 76–83. Association for Computational Linguistics.
- Massimo Poesio, Jon Chamberlain, Udo Kruschwitz, Livio Robaldo, and Luca Ducceschi. 2013. [Phrase detectives: Utilizing collective intelligence for internet-scale language resource creation](#). *ACM Transactions on Intelligent Interactive Systems*, 3(1).
- Massimo Poesio, Jon Chamberlain, Silviu Paun, Juntao Yu, Alexandra Uma, and Udo Kruschwitz. 2019. [A crowdsourced corpus of multiple judgments and disagreement on anaphoric interpretation](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1778–1789. Association for Computational Linguistics.
- Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. [CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes](#). In *Proceedings of the Shared Task: Modeling Multilingual Unrestricted Coreference in OntoNotes, Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1–40. Association for Computational Linguistics.
- Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. [Learning from crowds](#). *Journal of Machine Learning Research*, 11(43):1297–1322.
- Marta Recasens, Ed Hovy, and M. Antònia Martí. 2011. [Identity, non-identity, and near-identity: Addressing the complexity of coreference](#). *Lingua*, 121(6):1138–1152.
- Marta Recasens, M. Antònia Martí, and Constantin Orasan. 2012. [Annotating near-identity from coreference disagreements](#). In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 165–172. European Language Resources Association.
- Filipe Rodrigues and Francisco C. Pereira. 2018. [Deep learning from crowds](#). In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1611–1618.
- Willibald Ruch. 2008. [Psychology of humor](#). In Victor Raskin, editor, *The Primer of Humor Research*, number 8 in Humor Research, pages 17–100. Mouton de Gruyter, Berlin.
- Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. 2008. [LabelMe: A database and Web-based tool for image annotation](#). *International Journal of Computer Vision*, 77:157–173.
- Viktoriia Sharmanska, Daniel Hernández-Lobato, José Miguel Hernández-Lobato, and Novi Quadrianto. 2016. [Ambiguity helps: Classification with disagreements in crowdsourced annotations](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2194–2202.
- Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. 2008. [Get another label? Improving data quality and data mining using multiple, noisy labelers](#). In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622.
- Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. [Deep inside convolutional networks: Visualising image classification models and saliency maps](#). *CoRR*, abs/1312.6034.
- Edwin Simpson, Erik-Lân Do Dinh, Tristan Miller, and Iryna Gurevych. 2019. [Predicting humorousness and metaphor novelty with Gaussian process preference learning](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*,

pages 5716–5728. Association for Computational Linguistics.

Edwin Simpson and Iryna Gurevych. 2020. [Scalable Bayesian preference learning for crowds](#). *Machine Learning*, 109(4):689–718.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1994. [Inferring ground truth from subjective labelling of venus images](#). In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, page 1085–1092. MIT Press.

Alexandra Uma, Tommaso Fornaciari, Dirk Hovy, Silviu Paun, Barbara Plank, and Massimo Poesio. 2020. [A case for soft-loss functions](#). In *Proceedings of the 8th AAAI Conference on Human Computation and Crowdsourcing*, pages 173–177.

Alexandra Uma, Tommaso Fornaciari, Dirk Hovy, Silviu Paun, Barbara Plank, and Massimo Poesio. n.d. Learning from disagreements. Submitted.

Yannick Versley. 2008. [Vagueness and referential ambiguity in a large-scale annotated corpus](#). *Research on Language and Computation*, 6(3):333–353.

Sergey Zagoruyko and Nikos Komodakis. 2016. [Wide residual networks](#). *CoRR*, abs/1605.07146.

# SemEval-2021 Task 10: Source-Free Domain Adaptation for Semantic Processing

Egoitz Laparra<sup>1</sup>

Xin Su<sup>1</sup>

Yiyun Zhao<sup>1</sup>

Özlem Uzuner<sup>2</sup>

Timothy A Miller<sup>3</sup>

Steven Bethard<sup>1</sup>

<sup>1</sup>University of Arizona, Tucson, AZ 85721, USA  
{laparra, xinsu, yiyunzhao, bethard}@email.arizona.edu

<sup>2</sup>George Mason University, Fairfax, VA 22030, USA  
ouzuner@gmu.edu

<sup>3</sup>Boston Children’s Hospital and Harvard Medical School, Boston, MA 02115, USA  
timothy.miller@childrens.harvard.edu

## Abstract

This paper presents the Source-Free Domain Adaptation shared task held within SemEval-2021. The aim of the task was to explore adaptation of machine-learning models in the face of data sharing constraints. Specifically, we consider the scenario where annotations exist for a domain but cannot be shared. Instead, participants are provided with models trained on that (source) data. Participants also receive some labeled data from a new (development) domain on which to explore domain adaptation algorithms. Participants are then tested on data representing a new (target) domain. We explored this scenario with two different semantic tasks: negation detection (a text classification task) and time expression recognition (a sequence tagging task).

## 1 Introduction

Data sharing restrictions are common in NLP datasets. For example, Twitter policies do not allow sharing of tweet text, though tweet IDs may be shared. The situation is even more common in clinical NLP, where patient health information must be protected, and annotations over health text, when released at all, often require the signing of complex data use agreements.

The Source-Free Domain Adaptation shared task presents a new framework that asks participants to develop semantic annotation systems in the face of data sharing constraints. A participant’s goal is to develop an accurate system for a target domain when annotations exist for a related domain but cannot be distributed. Instead of annotated training data, participants are given a model trained on the annotations. Then, given unlabeled target domain data, they are asked to make predictions. This is

a challenging setting, and much previous work on domain adaptation does not apply, as it assumes access to source data (Ganin et al., 2016; Ziser and Reichart, 2017; Saito et al., 2017; Ruder and Plank, 2018), or assumes that labeled target domain data is available (Daumé III, 2007; Xia et al., 2013; Kim et al., 2016; Peng and Dredze, 2017).

Two different semantic tasks in English were created to explore this framework: negation detection and time expression recognition. These represent two common types of classification tasks: negation detection is typically formulated as predicting an attribute of a word or span given its context, and time expression recognition is typically formulated as a named entity tagging problem. Both of these tasks have previously been run as shared tasks, and had at least two different domains of data available, and we had access to experienced annotators for both tasks, allowing us to annotate data in a new domain.

Negation detection is the task of identifying negation cues in text. This task has been widely studied by previous work (Chapman et al., 2007, 2001; Harkema et al., 2009; Sohn et al., 2012) including the development of a variety of datasets (Uzuner et al., 2011; Mehrabi et al., 2015). However, there are still large performance losses in the cross-domain setting (Wu et al., 2014).

For negation detection, we provided a “span-in-context” classification model, fine-tuned on instances of the SHARP Seed dataset of Mayo Clinic clinical notes, which the organizers have access to but cannot currently be distributed. (Models were approved to be distributed, as the data is de-identified.) In the SHARP data, clinical events are marked with a boolean polarity indicator, with values of either asserted or negated. As development



	Source Collection	Source Domain	Instances	Negated instances
train	SHARP Seed	Mayo Clinic clinical notes	10,259	902
dev	i2b2 2010	Partners HealthCare clinical notes	5,545	1,115
test (unlabeled)	MIMIC III	Beth Israel ICU progress notes	622,703	-
test (labeled)	MIMIC III	Beth Israel ICU progress notes	9,580	958

Table 1: Size of the negation detection datasets. The train set is never distributed to the participants.

	Source Collection	Source Domain	Documents	Time entities
train	THYME	Mayo Clinic clinical notes	278	18,020
dev	TimeBank	News	99	2,231
test (unlabeled)	-	Food security	47	-
test (labeled)	-	Food security	17	1,900

Table 2: Size of the time expression recognition datasets. The train set is never distributed to the participants.

data, we used the i2b2 2010 Challenge Dataset, a de-identified dataset of notes from Partners HealthCare. The evaluation dataset for this task consisted of de-identified intensive care unit progress notes from the MIMIC III corpus (Johnson et al., 2016).

Time expression recognition has been a key component of previous temporal language related competitions, like TempEval 2010 (Pustejovsky and Verhagen, 2009) and TempEval 2013 (UzZaman et al., 2013). For this task, we followed the Compositional Annotation of Time Expressions (SCATE) schema (Bethard and Parker, 2016) used in SemEval 2018 Task 6 (Laparra et al., 2018). As in negation detection, previous works have also observed a significant performance degradation on domain shift (Xu et al., 2019).

For time expression recognition, we provided a sequence tagging model, fine-tuned on de-identified clinical notes from the Mayo Clinic, which were available to the task organizers, but are difficult to gain access to due to the complex data use agreements necessary. (Models were approved to be distributed, as the data is deidentified.) The development data was the annotated news portion of the SemEval 2018 Task 6 data whose source text is from the freely available TimeBank. For evaluation, we used a set of annotated documents extracted from food security warning systems.

The main impact of this task is to drive the NLP community to address the serious challenges of data sharing constraints by designing new domain adaptation algorithms that allow source data and target data to remain separate, rather than assuming they can be shared freely with each other.

## 2 Data and Resources

In this section, we describe both negation detection and time expression recognition tasks, the models fine-tuned on a difficult-to-obtain set of annotated data, the development data representing a new domain on which participants can explore their approaches for domain adaptation, and the test data representing another new domain on which the systems developed by participants are evaluated. Details of the different data sets can be found in Tables 1 and 2.

### 2.1 Negation detection

The negation detection track asks participants to classify clinical event mentions (e.g., diseases, symptoms, procedures, etc.) for whether they are being negated by their context.

For example, the sentence:

- (1) *Has no diarrhea and no new lumps or masses*

has three relevant events (diarrhea, lumps, masses), two cue words (both *no*), and all three entities are negated. This task is important in the clinical domain because it is common for physicians to document negated information encountered during the clinical course, for example, when ruling out certain elements of a differential diagnosis.

This task can be treated as a “span-in-context” classification problem, where the model jointly considers both the event to be classified and its surrounding context. For example, a typical transformer-based encoding of this problem for the *diarrhea* event in the example above looks like:

- (2) *Has no <e> diarrhea </e> and no new lumps or masses .*

**Pre-trained model** Participants were provided with a “span-in-context” classification model, trained on the 10,259 instances (902 negated) in the SHARP Seed dataset of de-identified clinical notes from Mayo Clinic, which the organizers had access to but cannot currently be distributed. In the SHARP data, clinical events are marked with a boolean polarity indicator, with values of either ASSERTED or NEGATED.

**Development data** Participants could use as development data the i2b2 2010 Challenge Dataset, a de-identified dataset of notes from Partners Health-Care, containing 5,545 entities labeled with an assertion status in the set {ASSERTED, NEGATED, UNCERTAIN, HYPOTHETICAL, CONDITIONAL, FAMILYRELATED}. We provided scripts that extracted i2b2 entities and simplified the label set to {NEGATED, NOTNEGATED}. Since the i2b2 2010 dataset consisted of notes from two sources, Partners and MIMIC III, the latter of which overlaps with our proposed test set, our script also filtered the development instances to contain only those from the Partners notes.

**Test data** During the testing period, participants were provided with the raw text of 622,703 instances drawn from the MIMIC III corpus<sup>1</sup>, which contains manually de-identified progress notes for patients from the intensive care unit of Beth Israel Deaconess Medical Center, with entities of interest already identified. From this, we manually annotated 9,580 instances of which 958 were negated.

## 2.2 Time expression recognition

The time expression recognition track, which represents a sequence-tagging task, uses the fine-grained time expression annotations that were a component of SemEval 2018 Task 6 (Laparra et al., 2018). For example:

- (3) In 

MONTH-OF-YEAR
January

 of 

YEAR
2009

, she experienced acute onset lower abdominal pain 

NUMBER
four to five

PERIOD
hours

AFTER
after

 her meal.

This task can be treated as a sequence classification problem, as in other named-entity tagging tasks.

<sup>1</sup><https://mimic.physionet.org/>

**Pre-trained model** Participants were provided with a sequence tagging model, trained on the 18,020 time expressions in the clinical portions of the SemEval 2018 Task 6, that were available to the task organizers, but are currently difficult to gain access to due to the complex data use agreements.

**Development data** Participants could use as development data the annotated news portion of the SemEval 2018 Task 6 data. The source text is from the freely available TimeBank<sup>2</sup>, and the 2,231 time entity annotations were from the freely available SCATE GitHub repository<sup>3</sup>.

**Test data** During the testing period, participants were provided with the raw text of 47 reports drawn from food security warning systems<sup>4</sup> and asked to predict time expressions. From this, we used 17 documents that included 1,900 time entities, annotated by two independent annotators and an adjudicator.

## 3 Evaluation Metrics

Negation detection was evaluated using the precision/recall/ $F_1$  of the negated class, as used in most published work. Time expression recognition was evaluated using the standard precision/recall/ $F_1$  previously used for the entity-finding portion of SemEval 2018 Task 6.

In both cases, the metrics are defined as:

$$P(S, H) = \frac{|S \cap H|}{|S|}$$

$$R(S, H) = \frac{|S \cap H|}{|H|}$$

$$F_1(S, H) = \frac{2 \cdot P(S, H) \cdot R(S, H)}{P(S, H) + R(S, H)}$$

where  $S$  is the set of items predicted by a system and  $H$  is the set of items manually annotated by humans.

## 4 Baseline Systems

To provide a comparison benchmark, we proposed two baselines for both negation detection and time expression recognition:

<sup>2</sup><https://www.cs.york.ac.uk/semeval-2013/task1/index.php%3Ffid=data.html>

<sup>3</sup><https://github.com/bethard/anafora-annotations>

<sup>4</sup>Like the UN World Food Programme <https://www.wfp.org/> or the Famine Early Warning Systems Network <https://fews.net/>.

Sub-task	System	source			dev			test		
		<i>P</i>	<i>R</i>	<i>F</i> <sub>1</sub>	<i>P</i>	<i>R</i>	<i>F</i> <sub>1</sub>	<i>P</i>	<i>R</i>	<i>F</i> <sub>1</sub>
Negation	Src-Trained	-	-	0.820	0.851	0.818	0.834	0.917	0.516	0.660
Negation	Dev-Tuned	-	-	-	-	-	-	0.908	0.611	0.730
Time Expression	Src-Trained	0.967	0.968	0.968	0.775	0.768	0.771	0.849	0.746	0.794
Time Expression	Dev-Tuned	-	-	-	-	-	-	0.827	0.782	0.804

Table 3: Performance of the baselines on the source domain, where **Source-Trained** (*Src-Trained*) was trained, and the two target domains (dev and test). For **Dev-Tuned**, dev set was also used for training.

**Source-Trained** Models pre-trained on only the source train data, i.e., the models that the organizers shared with the participants as explained in Section 2.

**Dev-Tuned** Models pre-trained on the source data (i.e., **Source-Trained**) and then fine-tuned on the labeled dev data.

All baselines were built on RoBERTa (Liu et al., 2019) using the HuggingFace Transformers library.<sup>5</sup>

Table 3 shows the performance of the baselines on negation detection and time expression recognition respectively. In both cases, there is a big drop in the performance of **Source-Trained** when it is applied to out-of-domain datasets. Using the development data to continue training the model (**Dev-Tuned**) provides some improvement for both tasks, but it is still far from in-domain performance.

## 5 Participating Systems

Since our goal was to see a set of experiments as varied as possible, we did not impose any constraint on the approaches participants could submit, including the use of any of the unlabeled or labeled data provided. The task had 9 participants that submitted 20 unique runs in total, as shown in Table 4. For each task, 2 submissions per team were allowed. There were 5 participants and 8 submission in **negation detection**, and 7 participants and 12 submissions in **time expression recognition**. Only 3 participants took part in both tasks.

### 5.1 Negation detection

*BLCUFIGHT-1* tried a self-training method fixing the top classifier so only the feature extractor was updated. Then, they ran an ensemble of 3 models. *BLCUFIGHT-2* built an unlabeled dataset selecting

<sup>5</sup><https://github.com/huggingface/transformers>.

2,000 instances from the development set, 2,000 from the test set and 2,886 from the training set. They used that unlabeled dataset progressively to continue fine-tuning the distributed model (for 2 epochs) following a self-learning approach. They additionally selected some negative prefixes and negative words as rules. The final predictions were obtained from an ensemble of 5 models.

*UArizona-1* used the development data to continue fine-tuning the distributed model (for 10 epochs). Then, they randomly sampled 3,000 examples from unlabeled test data and performed 2 self-learning iterations, using a 0.95 threshold to filter the pseudo training examples.

*IITK-1* also adapted the model with pseudo labels obtained from a sample of 25,000 instances from the test data. They selected predictions with low entropy as the pseudo training examples, performed data-augmentation on the selected instances, and used the resulting set to continue training the distributed model. *IITK-2* applied an adaptive version of this approach by slowly increasing the entropy threshold after each epoch and filtering again the training instances.

*MedAI-1* and *MedAI-2* followed a self-learning strategy preceded by a negation-aware pre-training process. For the latter, they built a dataset applying some heuristics on the test data. First, they manually collected a dictionary including negation cues, such as “not”, “no”, “no longer”. Second, they selected the nouns within a 3 token window around occurrences of the negation cues. Finally, they labeled the cue-noun pairs as negated instances.

**Observations:** Self-learning was the most widely applied technique (6 out of 8 submissions). 3 submissions extended this with heuristics, 2 submissions extended it with data augmentation, and 2 applied it with a model ensemble. Only 2 submissions leveraged the development set of which only 1 used the labeled data. All the

submission	task	dev data	test data	annotation	other	main technique
BLCUFIGHT-1	neg.	No	No	No	No	sf-train + ens
BLCUFIGHT-2	neg.	Unlabeled	Yes	No	No	sf-learn + heur + ens
UArizona-1	neg.	Labeled	Yes	No	No	sf-learn
IITK-1	neg.	No	Yes	No	No	sf-learn + dt-augm
IITK-2	neg.	No	Yes	No	No	sf-learn + dt-augm
MedAI-1	neg.	No	Yes	Heuristics	No	neg-train + sf-learn
MedAI-2	neg.	No	Yes	Heuristics	No	neg-train + sf-learn
Boom-1 <sup>†</sup>	neg.	-	-	-	-	-
BLCUFIGHT-1	time	Unlabeled	Yes	No	No	teach + sf-learn + heur + ens
BLCUFIGHT-2	time	Unlabeled	Yes	No	No	teach + sf-learn + heur
Self-Adapter-1	time	No	Yes	No	No	sf-learn
Self-Adapter-2	time	No	Yes	No	No	sf-learn
PTST-UoM-1	time	Labeled	Yes	No	No	sf-learn
YNU-HPCC-1	time	Labeled	No	No	No	train in dev + ens
YNU-HPCC-2	time	Labeled	No	No	No	train in dev + ens
UArizona-1	time	No	Yes	Manual	Yes	act-learn + dt-augm
UArizona-2	time	No	Yes	Manual	Yes	act-learn + dt-augm
KISNLP-1	time	Labeled	No	No	No	train in dev + dt-augm
KISNLP-2	time	Labeled	No	No	No	train in dev + dt-augm
Boom-1 <sup>†</sup>	time	-	-	-	-	-

<sup>†</sup> We did not receive feedback for these submissions.

Table 4: Some details on the tasks submissions. For each submission, the table reflects the **task** (*neg.* stands for negation) where it participates, if it uses the *unlabeled* or *labeled* development data (**dev data**), if it uses the *unlabeled test data*, if participants carried out some manual or heuristics-based **annotation**, if **other** source of data is used and the **main techniques** applied. List of abbreviations in the *main technique* column: *act-learn* for active learning, *dt-augm* for data augmentation, *ens* for ensemble, *heur* for heuristics, *neg-train* for negation-aware pre-training, *sf-learn* for self learning, *sf-train* for self training, *teach* for mean teacher.

submissions but one used the unlabeled test data to produce a training set for the target domain, either in the form of pseudo-labeled instances (5 submissions) or by heuristic-driven annotation (2 submissions). No submissions used additional resources.

## 5.2 Time expression recognition

*BLCUFIGHT-1* and *BLCUFIGHT-2* proposed an unsupervised mean-teacher framework that updates the model in a self-learning manner. Additionally, they used a set of string-matching heuristics derived from the development set, e.g., “spring” or “summer” for Season-Of-Year, and “decades” for Period. *BLCUFIGHT-1* ensembled 2 models for a better robustness.

*Self-Adapter-1* and *Self-Adapter-2* generated pseudo training examples by running the provided model on the test documents and selecting the sentences where the highest words’ entropy was lower than 0.1. In *Self-Adapter-1*, they combined the

predictions of both a fixed version and a trainable version of the model. *Self-Adapter-2* used only the trainable model. In both submissions, the trainable model was updated by applying 3 iterations of the *sloughing trick*, i.e., training the model iteratively with the pseudo-labels obtained by the model of the previous iteration.

*PTST-UoM-1*, also following a self-training approach, built, for each unlabeled input sentence, a chart containing high probability label sequences produced by the distributed model and applied it as a supervision signal. They used the labeled development data for tuning some of the hyperparameters.

*UArizona-1* combined active learning and data augmentation. They ran 5 iterations of the following steps: 1) predict the unlabeled test data and then select 32 sentences with high entropy calculated as the sum of the entropy of all tokens in the sentence; 2) manually label time entities in the 32 sentences; 3) for each manually labeled time entity, generate

5 additional training examples using 5 new words with same entity type; 4) train the model on the resulting dataset. The same method was used by *UArizona-2*, but, in this case, they fixed some errors in the manual annotations.

*KISNLP-1* and *KISNLP-2* used the development labeled data as a fine-tuning resource, which was complemented by a data augmentation process. They did not use the unlabeled test data, nor any other resource.

*YNU-HPCC-1* and *YNU-HPCC-2* also used the labeled portion of the development set. They fine-tuned 4 popular transformer-based pre-trained models: RoBERTa, BERT (Devlin et al., 2019), DistilBERT (Sanh et al., 2020) and ALBERT (Lan et al., 2020). The final prediction was given by hard voting strategy, integrating the results of the 4 models along with **Source-Trained**.

**Observations:** Self-learning (5 submissions) and data augmentation (4 submissions) were the most commonly followed approaches. 2 submissions extended a self-learning technique with manually created heuristics. Only 3 submissions proposed ensemble methods. In this task, the development set was more frequently exploited and 4 submissions made use of the labeled data to continue fine-tuning the provided model. The test set was manually annotated by 2 submissions that followed an active learning approach, along with some additional resources. 4 submissions did not use the unlabeled test data.

## 6 Evaluation Results

Tables 5 and 6 shows the performance of the systems described in Section 5 on negation detection and time expression recognition. For comparison, the tables also include the performance of the baselines described in Section 4.

### 6.1 Negation detection

As shown in Table 5, 7 out of 8 submissions on negation detection outperform **Source-Trained** but only 4 performed better than **Dev-Tuned**.

The best results were obtained by *MedAI-1* and *MedAI-2*, achieving 16.2 and 9.2 percentage points of  $F_1$  more than **Source-Trained** and **Dev-Tuned**, respectively. These model had a large recall improvement (14.5 points more than **Source-Trained** and 24.0 more than **Dev-Tuned**) at the expense of a slight degradation in precision.

System	$P$	$R$	$F_1$
MedAI-1 <sup>†</sup>	0.902	<b>0.756</b>	<b>0.822</b>
MedAI-2 <sup>†</sup>	0.902	<b>0.756</b>	<b>0.822</b>
UArizona-1 <sup>+†</sup>	0.880	0.680	0.767
BLCUFIGHT-2 <sup>*†</sup>	0.913	0.616	0.736
IITK-2 <sup>†</sup>	0.876	0.624	0.729
Boom-1	0.929	0.597	0.727
IITK-1 <sup>†</sup>	<b>0.939</b>	0.566	0.706
BLCUFIGHT-1	0.528	0.639	0.578
Dev-Tuned	0.908	0.611	0.730
Source-Trained	0.917	0.516	0.660

Table 5: Official results (ranked by  $F_1$ ) on negation detection. Superscripts indicate that the submission used: \*unlabeled dev, +labeled dev or †unlabeled test data

*IITK-1* and *Boom-1* outperform both baselines in terms of precision but obtain a worse recall than **Dev-Tuned**.

The 3 best submissions on this task (*MedAI-1*, *MedAI-2* and *UArizona-1*) make use of some kind of labeled data. In the case of *MedAI-1* and *MedAI-2*, this data belongs to the target test domain, which could explain the good results of these 2 submissions. *BLCUFIGHT-2*, the next best performing system and the only other one that outperforms both baselines, also applies some manual supervision in the form of hand-crafted rules.

In general, self-learning proved to be an effective technique for negation detection, especially in terms of recall, while data-augmentation also shows recall improvements in some cases. As usual, ensemble models are helpful. Including some manual supervision drove the largest gains.

### 6.2 Time expression recognition

Table 6 shows that for time expression recognition, 9 out of 12 submissions outperformed **Source-Trained** and only 3 obtained a better performance than **Dev-Tuned**. The gains were generally smaller than on negation detection, with the best models being only 2.1 percentage points of  $F_1$  above **Source-Trained** and 1.1 percentage points above **Dev-Tuned**.

As in negation detection, the best performing system (*BLCUFIGHT-1*) utilizes some form of manual supervision. In this case, they apply a set of manually created string matching heuristics in combination with a *self-learning* approach that is boosted by a model ensemble.

In this task, the use of the labeled development

System	$P$	$R$	$F_1$
BLCUFIGHT-1* <sup>†</sup>	0.847	0.785	<b>0.815</b>
Self-Adapter-1 <sup>†</sup>	0.873	0.757	0.811
BLCUFIGHT-2* <sup>†</sup>	0.834	0.787	0.810
YNU-HPCC-2 <sup>+</sup>	0.817	0.791	0.803
Self-Adapter-2 <sup>†</sup>	0.839	0.760	0.797
PTST-UoM-1 <sup>++</sup>	<b>0.901</b>	0.713	0.796
UArizona-1 <sup>†</sup>	0.786	0.804	0.795
UArizona-2 <sup>†</sup>	0.783	<b>0.807</b>	0.795
Boom-1	0.869	0.732	0.795
KISNLP-1 <sup>+</sup>	0.810	0.777	0.793
KISNLP-2 <sup>+</sup>	0.798	0.764	0.781
YNU-HPCC-1 <sup>+</sup>	0.872	0.655	0.748
Dev-Tuned	0.827	0.782	0.804
Source-Trained	0.849	0.746	0.794

Table 6: Official results (ranked by  $F_1$ ) on time expression recognition. Superscripts indicate that the submission used: \*unlabeled dev, +labeled dev or <sup>†</sup>unlabeled test data

set is more frequent. 5 of the submissions made use of this data, but none obtained better results than **Dev-Tuned**, although *YNU-HPCC-2* got a close  $F_1$  score. In the case of *PTST-UoM-1*, this explained by the fact that they only consulted this set to fine-tune the hyperparameters of their model, although this strategy was enough to obtain the best precision among all systems. The approach of *KISNLP-1* and *KISNLP-2* is the same as **Dev-Tuned** but combined with some data-augmentation, resulting in a drop in performance. This may be caused by only using the development set to perform the augmentation since, after all, it belongs to a different domain than the test documents. *YNU-HPCC-2* is the only submission, along with *YNU-HPCC-1*, that utilized other pre-trained transformers, in an ensemble mode, besides the model provided.

*UArizona-1* and *UArizona-2* are the only submissions that tried an *active learning* strategy. The approach performed slightly better than **Source-Trained** but worse than **Dev-Tuned**. This contrasts with the best performing model on negation detection that also implemented a manual annotation process on test data, but it is explained by the much more complex annotation scheme of time expressions. *UArizona-2* obtains the best recall on the task.

*Self-Adapter-1* is the only submission that outperforms **Dev-Tuned** without using any kind of manual supervision. The only difference with re-

spect to *Self-Adapter-2*, that did not perform as well, is that the original model trained on the source domain is consulted to produce pseudo-examples in every iteration of their self-learning technique. This seems to counteract a possible degradation of the predictions caused by updating the model with pseudo-labels.

## 7 Future directions

Self-learning and data augmentation were the most frequently used techniques. Some systems, including the best performing ones, incorporated some kind of manual supervision in the form of active-learning, hand-crafted heuristics or semi-automatically building a training set. This suggests that future work on source-free domain adaptation will focus on acquiring data instances for the target domain either automatically or manually, and use such data to continue fine-tuning the source-domain model.

Any new approaches will have to address some fundamental challenges. Errors in the generation of pseudo-labels propagate in successive self-learning iterations degrading the performance. Continual fine-tuning on data from a new domain can lead to catastrophic forgetting, especially if the data is restricted to certain instances like those drawn from high-confident predictions of the source model. Manually supervised approaches, such as active learning, do not necessarily solve these problems due to the complexity of some annotation schemes, like in time expressions recognition, and the reduced number of labels that this methods can yield.

Some of the experiments carried out during this task have approached these issues and should be taken as an starting point for future research.

## 8 Conclusion

In this paper, we have described the Source-Free Domain Adaptation shared task held within SemEval-2021. In this task, participants were asked to adapt a given model to a target domain when the access to both labeled and unlabeled source data is restricted. In contrast to previous tasks on domain adaptation, participants were only provided with a trained model and the target unlabeled data. Systems were evaluated on two tasks, negation detection and time expression recognition, that are paradigmatic examples of two common types of machine-learning problems in natural language processing: text classification and sequence

labeling.

9 participants took part in the challenge with 20 different systems. In negation detection, 8 submissions were received from 5 participants while 7 participants submitted 12 runs for time expression recognition. 3 participants presented approaches for both tasks. 7 out of 8 submissions for negation detection and 9 out of 12 submissions for time expression recognition outperformed the model trained on the source domain. Compared to the same model fine-tuned on the development data, 4 systems in negation detection and 3 in time expression recognition showed a better performance.

This is the first time that such a framework is formally designed and aims to draw the community's attention to a challenging problem that seriously affects the deployment of NLP models to real-life scenarios, like health institutions.

The scripts and the code of the baselines, along with the development and test data, can be obtained from the task's GitHub repository.<sup>6</sup> The trained models are available in the HuggingFace model hub for both negation detection<sup>7</sup> and time expression recognition.<sup>8</sup> The CodaLab<sup>9</sup> leader-board of the of the post-evaluation phase will continue to accept submissions indefinitely.

## Acknowledgments

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under Award Numbers R01LM012918 and R01LM010090. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

## References

Steven Bethard and Jonathan Parker. 2016. [A semantically compositional annotation scheme for time normalization](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3779–3786, Portorož, Slovenia. European Language Resources Association (ELRA).

<sup>6</sup><https://github.com/Machine-Learning-for-Medical-Language/source-free-domain-adaptation>

<sup>7</sup>[https://huggingface.co/tmills/roberta\\_sfda\\_sharpseed](https://huggingface.co/tmills/roberta_sfda_sharpseed)

<sup>8</sup><https://huggingface.co/clulab/roberta-timex-semeval>

<sup>9</sup><https://competitions.codalab.org/competitions/26152>

Wendy Chapman, John Dowling, and David Chu. 2007. [ConText: An algorithm for identifying contextual features from clinical text](#). In *Biological, translational, and clinical language processing*, pages 81–88, Prague, Czech Republic. Association for Computational Linguistics.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. [A simple algorithm for identifying negated findings and diseases in discharge summaries](#). *Journal of Biomedical Informatics*, 34(5):301–310.

Hal Daumé III. 2007. [Frustratingly easy domain adaptation](#). In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. [Domain-adversarial training of neural networks](#). *Journal of Machine Learning Research*, 17(59):1–35.

Henk Harkema, John N. Dowling, Tyler Thornblade, and Wendy W. Chapman. 2009. [ConText: An algorithm for determining negation, experiencer, and temporal status from clinical reports](#). *Journal of Biomedical Informatics*, 42(5):839–851. Biomedical Natural Language Processing.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Young-Bum Kim, Karl Stratos, and Ruhi Sarikaya. 2016. [Frustratingly easy neural domain adaptation](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 387–396, Osaka, Japan. The COLING 2016 Organizing Committee.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A lite BERT for self-supervised learning of language representations](#). In *International Conference on Learning Representations*.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. [SemEval 2018 task 6: Parsing time normalizations](#). In *Proceedings*

- of *The 12th International Workshop on Semantic Evaluation*, pages 88–96, New Orleans, Louisiana. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M. Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C. Max Schmidt, Hongfang Liu, and Mathew Palakal. 2015. DEEPEN: A negation detection system for clinical text incorporating dependency relation into NegEx. *Journal of Biomedical Informatics*, 54:213–219.
- Nanyun Peng and Mark Dredze. 2017. Multi-task domain adaptation for sequence tagging. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 91–100, Vancouver, Canada. Association for Computational Linguistics.
- James Pustejovsky and Marc Verhagen. 2009. SemEval-2010 task 13: Evaluating events, time expressions, and temporal relations (TempEval-2). In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 112–116, Boulder, Colorado. Association for Computational Linguistics.
- Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, Melbourne, Australia. Association for Computational Linguistics.
- Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. 2017. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.
- S. Sohn, Stephen T Wu, and C. Chute. 2012. Dependency parser-based negation detection in clinical narratives. *AMIA Summits on Translational Science Proceedings*, 2012:1 – 8.
- Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.
- Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. SemEval-2013 task 1: TempEval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Stephen Wu, Timothy Miller, James Masanz, Matt Coarr, Scott Halgrim, David Carrell, and Cheryl Clark. 2014. Negation’s not solved: Generalizability versus optimizability in clinical natural language processing. *PLOS ONE*, 9(11):1–11.
- Rui Xia, Xuelei Hu, Jianfeng Lu, Jian Yang, and Chengqing Zong. 2013. Instance selection and instance weighting for cross-domain sentiment classification via PU learning. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI ’13*, pages 2176–2182. AAAI Press. Event-place: Beijing, China.
- Dongfang Xu, Egoitz Laparra, and Steven Bethard. 2019. Pre-trained contextualized character embeddings lead to major improvements in time normalization: a detailed analysis. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (\*SEM 2019)*, pages 68–74, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yftah Ziser and Roi Reichart. 2017. Neural structural correspondence learning for domain adaptation. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 400–410, Vancouver, Canada. Association for Computational Linguistics.



# BLCUFIGHT at SemEval-2021 Task 10: Novel Unsupervised Frameworks For Source-Free Domain Adaptation

**Weikang Wang, Yi Wu, Yixiang Liu, Pengyuan Liu**  
Beijing Language and Culture University, Beijing, China  
{978955719wwk, wyclover51}@gmail.com  
lyx\_blcu@163.com, liupengyuan@pku.edu.cn

## Abstract

Domain adaptation assumes that samples from source and target domains are freely accessible during a training phase. However, such assumption is rarely plausible in the real-world and may cause data-privacy issues, especially when the label of the source domain can be a sensitive attribute as an identifier. SemEval-2021 task 10 focuses on these issues. We participate in the task and propose novel frameworks based on self-training method. In our systems, two different frameworks are designed to solve text classification and sequence labeling. These approaches are tested to be effective which ranks the third among all systems in subtask A, and ranks the first among all systems in subtask B.

## 1 Introduction

Deep neural networks have achieved remarkable success in a variety of applications across different fields while with huge expense of laborious large-scale training data annotation. To avoid expensive data labeling, domain adaptation (DA) methods were proposed to fully utilize previously labeled datasets and unlabeled data on hand in a transductive manner, which obtained promising results in sentiment analysis, part-of-speech tagging, machine translation, etc. (Glorot et al., 2011; Yang and Eisenstein, 2014; Chu and Wang, 2018)

Unsupervised Domain Adaptation (UDA) aims to reduce the domain shift between labeled and unlabeled target domains. Early works (Blitzer et al., 2006; Pan et al., 2010) learnt domain-invariant features to link the target domain to the source domain. Along with the growing popularity of deep learning, plenty of works benefited from its powerful representation learning ability for domain adaptation. Those methods typically minimized the distribution discrepancy between two domains (Plank et al., 2014), or deployed adversarial training (Ganin and Lempitsky, 2015; Bousmalis et al., 2016; Li et al., 2018).

However, a crucial requirement in the methodology of these methods is that all samples from both domains are freely available during the training process, which is inefficient in data transmission and may violate the data privacy policy. For example, it is not allowed to share tweet texts according to Twitter policies, though tweet IDs can be shared. The situation is even more common in clinical NLP, where patient health information must be protected, and annotations over health text, when released at all, often require the signing of complex data use agreements.

SemEval 2021 task 10 focuses on the problem of source-free domain adaptation for semantic processing. Subtask A of task 10 is negation detection which aims to classify clinical event mentions (e.g., diseases, symptoms, procedures) for whether they are being negated by their context. Traditional systems, such as one of the first algorithms NegEx (Chapman et al., 2001) was based on rules. Subsequently, syntax-based methods were developed (Huang and Lowe, 2007; Mehrabi et al., 2015). In recent years, some researchers explored new generation of transfer learning models (BERT) to solve this task (Khandelwal and Sawant, 2019), outperforming the previous state-of-the-art systems by a significant margin.

Subtask B of task 10 is time expression recognition which aims to find time expressions in text. It is a sequence labeling task as (Laparra et al., 2018) described in their work. A few of works combined traditional machine learning with rules achieved good performances (Olex et al., 2018). Some studies got character-level contextual embeddings (Xu et al., 2019) and applied to this task, yielding major performance improvements over the previous state-of-the-art.

In this paper, we propose two different unsupervised frameworks for each subtask in source-free setting. For negation detection task, we design a framework which obtains pseudo labels with high

confidence by using reliable pseudo labels as prototypes. For time expression recognition, we design an unsupervised teacher-student framework with Mean Teacher.

## 2 System description

For subtask A, we used a pseudo-labeling training method. To reduce the uncertainty from the pseudo labels, we only chose those with high confidence to fine-tune the model. Finally, we ensembled 5 models to make the model have better robustness and results. For subtask B, we started by data pre-processing. Then, we enlarged the training set with pseudo-labeled sentences, which were predicted on the test set by teacher model. In addition, Mean Teacher helps to generate better pseudo labels. Finally, we used the ensemble model to make predictions and add manual expressions. Each module will be introduced in detail in the following sections.

### 2.1 subtask A: Negation detection

#### 2.1.1 Pre-processing

Samples in the test data was split by punctuation to a single sentence which included the entity being detected. This was done to avoid the impact of the irrelevant context. All white spaces were removed.

#### 2.1.2 Architecture

For negation detection, we utilized the RoBERTa-base (Liu et al., 2019) pretrained model fine-tuned on the 10,259 instances (902 negated) in the SHARP Seed dataset which is different from the target domain. To adapt the source domain to the target domain, we kept the feature extractor of the source model fixed and trained the classifier module by using pseudo labels with high confidence (He and Zhou, 2011). It aims to learn a domain-specific classifier learning module.

Our model is composed of two parts, the first part is Adaptive Prototype Memory (APM) (Kim et al., 2020), which provides pseudo labels with high confidence for the target model. The second part is the target model where parameters of the feature extractor are fixed, i.e., does not participate in training. The overall architecture of our model is shown in Fig.1

**Pseudo Labeling:** Pseudo labeling (Lee et al., 2013) was originally proposed for semi-supervised learning. Since Pseudo labeling is a simple and efficient method, it gains popularity in other trans-

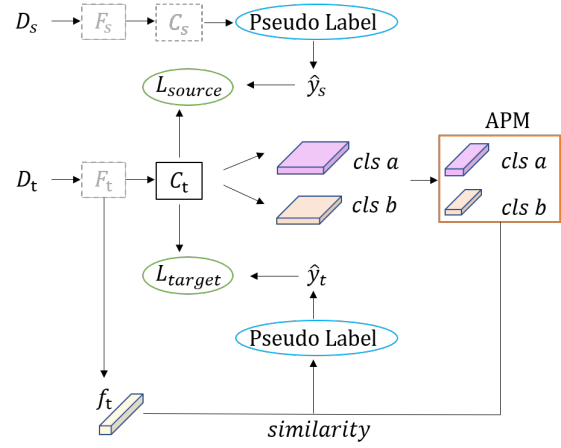


Figure 1: Overall flow of subtask A framework. In the figure, **D**, **F**, **C** and **L** represent Data, Feature extractor, Classifier and Loss, respectively. The subscripts **s** and **t** indicate whether they come from the source domain or the target domain. Dashed lines indicate fixed model parameters.

ductive learning problems like **Domain Adaptation**. The main idea is to label unlabeled data with the maximum predicted probability and perform fine-tuning together with labeled data. For this task, we don't have labeled training data, so our method uses a more reliable pseudo-labels to fine-tune the model.

**APM:** To obtain reliable pseudo labels, prediction uncertainty is measured by self-entropy, i.e.,  $H(x) = -\sum p(x)\log(p(x))$ . The smaller the entropy is, the more confidence of the prediction is. First of all, we calculated the normalized self-entropy of target samples.

$$H(x_t) = -\frac{1}{\log N_c} (x_t) \log(l(x_t))$$

where  $N_c$  refers to the number of classes,  $l(x_t)$  is the output of the target classifier,  $x_t$  represents the samples from the target domain. The next step is to select the reliable part among all target samples, i.e., the part with smaller entropy. In order to minimize the influence of incorrect pseudo labels, we chose 20% as a threshold to get reliable samples. So the top 20% target samples of the smaller entropy are stored in the APM.

Based on prototypes from APM module which can represent each class, we can assign labels to unlabeled target data according to similarity score:

$$S(x_t) = \frac{1}{|M_c|} \sum_{p_c \in M_c} \frac{p_c^T f_t}{\|p_c\|_2 \|f_t\|_2}$$

where  $c$  represents two classes, i.e., “negated” or “not negated”,  $f_t$  and  $p_c$  stand for the embedded feature of target data and prototype respectively.

**Loss Function:** Pseudo labels generated by the first part are used to train the classifier of the target model. During the training process, to avoid the influence of unstable pseudo labels, our loss consists of two parts. One is the loss of the source classifier, and the other is the loss of the trainable target classifier.

$$L_{total}(D_t) = (1 - \alpha)L_{source}(D_t) + \alpha L_{target}(D_t)$$

At the beginning, loss of the source classifier accounts for a large proportion, it is added for regularization, because the generated label may be unstable. With the increase of training steps, the proportion of source decreases gradually, while the proportion of the loss of the target classifier increases.

### 2.1.3 Ensembling

To obtain a more robust model, we trained five models by changing the hyper parameters, and integrated the five models by voting ensemble method. Test data were passed through the ensembled model as the final output of the system.

## 2.2 Subtask B: Time expression recognition

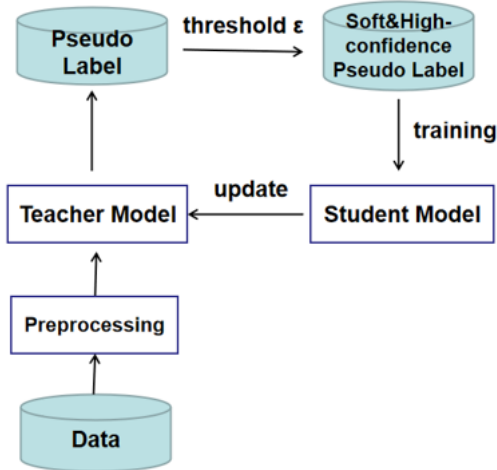


Figure 2: Overall flow of subtask B framework

### 2.2.1 Pre-processing

- Denoising: Since the first two lines of the development set text are the descriptions of the file name and would not appear in the test set, we removed the first two lines of all verification set texts.

- Training set: Development set and randomly selected partial test set.

### 2.2.2 Teacher and Student architecture

The main part of our system for subtask B is the teacher-student framework (Liang et al., 2020), which is an unsupervised method. Specifically, The teacher model is initialized by student model. To avoid losing too much information of other classes, we proposed to use soft labels. Recall that for the  $n$ -th token in the  $m$ -th sentence, the output probability simplex over  $C$  classes is denoted as:

$$[f_{n,1}(X_m; \theta), \dots, f_{n,C}(X_m; \theta)].$$

After the teacher model generated soft labels from training set (let denote  $\{S_m = [s_{m,n}]_{n=1}^N\}_{m=1}^M$  the soft pseudo-labels generated from teacher model), in order to further address the uncertainty in the data, we selected tokens based on the prediction confidence. That is to say, we selected a set of high confidence tokens from the  $m$ -th sentence by

$$H_m = \{n : \max_c s_{m,n,c} > \epsilon\},$$

where  $\epsilon \in (0, 1)$  is a tuning threshold. The high confidence selection essentially enforces the student model to better fit tokens with high confidence, and therefore is able to improve the model robustness against low-confidence tokens. Loss1 is denoted as:

$$Loss1 = \frac{1}{M} \sum_{m=1}^M \ell_{KL}(S_m^{(t)}, f(X_m; \theta))$$

where  $\ell_{KL}(\cdot, \cdot)$  denotes the KL-divergence-based loss:

$$\ell_{kl}(S_m, f(X_m; \theta)) = \frac{1}{|H_m|} \sum_{n \in H_m} \sum_{c=1}^C -s_{m,n,c} \log f_{n,c}(X_m; \theta).$$

### 2.2.3 Mean Teacher

In our architecture, we also added Mean Teacher loss to update student model. Mean Teacher (Tarvainen and Valpola, 2017) is a simple but effective method to improve teacher model performance. After the weights of the student model have been updated with gradient descent, the teacher model weights are updated as an moving average of the student weights as follows:

$$\theta'_t = \alpha\theta'_{t-1} + (1 - \alpha)\theta_t,$$

where  $\alpha$  is a smoothing coefficient hyperparameter. Loss2 denote Mean Teacher loss (same as the formula in 2.2.2), but  $\ell_{KL}(\cdot, \cdot)$  denotes:

$$\ell_{kl}(S_m, f(X_m; \theta)) = \frac{1}{N} \sum_{n=1}^N \sum_{c=1}^C -s_{m,n,c} \log f_{n,c}(X_m; \theta).$$

Accordingly, the student model can be optimized by minimizing the loss (consists of Loss1 and Loss2).

The process described above is repeated periodically to train the student model. Eventually, early stopping is adapted to prevent student model from overfitting.

#### 2.2.4 Post-processing

- **Ensembling:** Ensemble has shown its power on effectively improving the robustness and accuracy of each individual prediction (Opitz and Maclin, 1999; Rokach, 2010). By ensembling predictions from models with different hyper-parameters or architectures, we can get better results than each individual model. In our system, we set different hyper-parameters on learning rate and random seed. In this case, two ensemble model generate predictions individually and we take the union of the two independent predictions as model predictions.
- **Manual rules:** Through observation on the data, we obtained a set of feature words which appear frequently. Specifically, we labeled feature words “daily” and “annual” with “Calendar-Interval”, label “minutes” and “decades” with “Period”, label “ago” and “before” with “Before” etc.

### 3 Experiments

#### 3.1 Data

SemEval 2021 Task 10 released the training, development and test dataset.

For subtaskA, the development data is the i2b2 2010 Challenge Dataset, a de-identified dataset of notes from Partners HealthCare, containing 2886 unlabeled train instances (entities in sentence context), and 5545 dev instances with a corresponding labeling for with negation status. The test data is from the MIMIC III corpus v1.4, which is much

	Train	Dev	Test
Positive	-	1115	958
Negative	-	4430	8622
Total	2886	5545	9580

Table 1: Data distribution of subtask A.

messier than the development data. The detailed statistics are shown in Table 1.

For subtask B, we found that the category labels were severely imbalanced. Specifically, the training set and the dev set have label types which are not mutually exclusive. In addition, the dev set labels is mainly distributed in Month-Of-Year, Day-Of-Month, Period, etc., while the test set labels are mainly distributed in Month-Of-Year, Season-Of-Year, Year, etc.

#### 3.2 Evaluation Metrics

F1, Precision and Recall were used to evaluate the performance of both subtask A and subtask B. The evaluation will verify whether the predicted “label” is the same as the desired “label” which is annotated by human workers, and then calculate its F1 scores, precision and recall.

#### 3.3 Experimental Details

**Hyper-Parameters of subtask A.** Since we were training the model with unlabeled data, we added the same amount of dev and test data as train data to fine-tune the model to get better results. We use an Adam optimizer to tune the parameters with learning rate = 5e-5, max seq length = 128, batch size = 32, seed = 40 and we trained each model for 2 epochs. Then we used the APM module to get  $M = 400$  prototypes which represents each class. By computing the similarity between each target sample and all prototypes in APM, we obtained 8658 pseudo-labels with high confidence.

**Hyper-Parameters of subtask B.** For subtask B, we trained our two ensemble model (each with three models) on unlabeled data with seed = 32,42, learning rate = 2e-5,2.5e-5,3e-5, we trained each model for 5 epochs with early stopping. We also used an AdamW optimizer to tune the parameters with epsilon = 1e-6, batch\_size = 16.

### 4 Results

The performance of our system and the task baselines for both subtasks are shown in following tables.

Model	F1	Precision	Recall
Baseline	0.660	0.917	0.516
Baseline(fine-tuned)	0.730	0.908	0.611
SFDA w/o $L_{source}$	0.674	0.874	0.548
SFDA w/o APM	0.686	0.927	0.545
SFDA	0.717	<b>0.936</b>	0.581
SFDA+ensemble	<b>0.736</b>	0.913	<b>0.616</b>

Table 2: Results of different ablation experiments for subtask A. All the models are trained on training data, development data and test data.

Table 2 shows the results of several ablation experiments. Compared with models without APM or  $L_{source}$ , we found that adding both together improved the performance of the model. The voting ensemble model of single models trained with extra development and test data outperforms all other models and achieves the highest F1 score.

Model	F1(dev)	F1(test)
SFDA(t) w/o $L_{source}$	0.838	0.661
SFDA(t) w/o APM	0.814	0.717
SFDA(t)	0.859	0.707
SFDA(t)+ensemble	<b>0.873</b>	0.720
SFDA(t+d) w/o $L_{source}$	0.864	0.689
SFDA(t+d) w/o APM	0.851	0.668
SFDA(t+d)	0.868	0.718
SFDA(t+d)+ensemble	0.870	<b>0.725</b>

Table 3: Results of models trained on different data sets. SFDA(t) refers to the model trained on train data and SFDA(t+d) represents the model trained on both train data and development data.

Table 3 shows the results of models trained on different data sets. Since we don't need labeled data to train the model, we added development and test data to train the model. Compared with our final model which was trained on three data sets, models trained on fewer data sets, i.e., only on train data or on both train data and development data perform less well.

Figures 3 and 4 show the confusion matrix of the classification results of baseline model and our model on the test dataset. This corresponds to *Baseline* and *SFDA+ensemble* in Table2, respectively. Compared with baseline, we predicted more true positive samples. However, the false prediction of negative samples as positive has increased. As a result, the recall and F1 score of our model have been improved, but the precision has decreased a little.

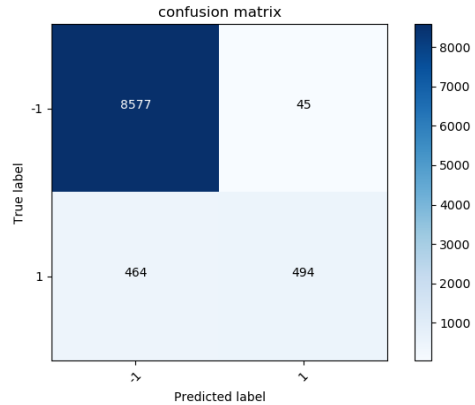


Figure 3: The confusion matrix of the baseline model

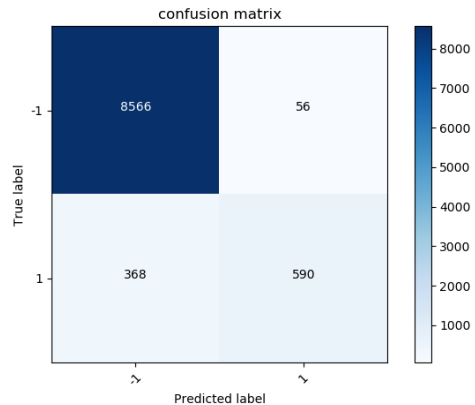


Figure 4: The confusion matrix of our model

Table 4 shows the results of several ablation experiments. Without soft labels, we can find that F1 score drop significantly. A possible explanation is that the soft labels preserve more information and generate better labels. Based on soft labels, MT and manual rules marginally improve the F1 scores. Finally, the ensemble model (MT+soft+rules) outperforms all other models and achieves the highest F1 score.

Table 5 shows the results of our model trained on dev set. Compared with the dev set, the F1 of the test set dropped by an average of 2%.

**Error analysis.** For subtask A, we conducted statistics and analysis on the classification results of Baseline model and our best model. There are 474 sentences that both Baseline model and our best model predict correctly. The entity being detected usually follow the word that express negative

<sup>1</sup>Model pre-trained on only the source data (official provided).

<sup>2</sup>Model pre-trained on the source data and then fine-tuned on the dev set (official provided).

<sup>3</sup>Here MT refer to Mean Teacher.

Model	F1	Precision	Recall
Baseline <sup>1</sup>	0.794	0.849	0.746
Baseline(fine-tuned) <sup>2</sup>	0.804	0.827	0.782
MT <sup>3</sup>	0.755	0.747	0.763
soft	0.807	0.859	0.761
MT+soft	0.801	0.854	0.754
MT+soft+rules	0.812	<b>0.863</b>	0.767
MT+soft+rules(ensemble)	<b>0.815</b>	0.847	<b>0.785</b>

Table 4: Results of different ablation experiments for subtask B. Our models are trained on training set.

Model	F1(dev)	F1(test)
Baseline	0.746	0.794
Baseline(fine-tuned)	-	0.804
MT	0.767	0.747
soft	0.815	0.791
MT+soft	0.813	0.799
MT+soft(ensemble)	<b>0.832</b>	<b>0.814</b>

Table 5: Results of subtask B. Our models are trained on dev set.

meanings in these sentences closely. e.g. "... *no* <e>erythema </e>...", "... *denies* <e>chest pain </e>..." ... There are 116 sentences that our model predicts correctly but the baseline predicts incorrectly. In this part, there are some long-distance keywords or parallel phrases. e.g. "... *No tobacco, EtOH, or* <e>IV drug use </e>". There are 348 sentences that are not predicted correctly by both models. For these, we consider to add some hand-craft rules to improve the results of the model.

For subtask B, we conducted a manual error analysis. For the raw text "*during the harvest season*", both our model and baseline model incorrectly labeled "harvest" with "Season-Of-Year" instead of "harvest season", "harvest" is just the activity, though if it instead said "harvest season", we would annotate that whole thing as a "Season-Of-Year". For the raw text "*February (27,661)*", baseline model incorrectly label "27" with "Day-Of-Month" while our model didn't, which proves the effectiveness of our architecture. In addition, we list a detailed description of the recall of subtask B in Table 6.

## 5 Conclusion

We introduce two different frameworks which are both based on self-training method for text classification and sequence labeling in SemEval 2021 task 10, in order to address the problems of source-

free, labeled training data scarcity. In subtask A, we used a metric learning method, combining pseudo labeling with prototype network and achieve good results. In subtask B, we employed teacher-student framework, and then we propose to use high-confidence soft labels to further improve the self-training. Our system takes third place in subtask A and first place in subtask B.

In future, we would like to introduce adversarial training and more data augmentation approaches in our model to further facilitate source-free domain adaptation.

## Acknowledgments

Thanks for Shucheng Zhu's suggestions on writing this paper. Pengyuan Liu is the corresponding author.

## References

- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128.
- Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. *arXiv preprint arXiv:1608.06019*.
- Wendy W Chapman, Will Bridewell, Paul Hanbury, Gregory F Cooper, and Bruce G Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, 34(5):301–310.
- Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. *arXiv preprint arXiv:1806.00258*.
- Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.
- Yulan He and Deyu Zhou. 2011. Self-training from labeled features for sentiment analysis. *Information Processing & Management*, 47(4):606–616.
- Yang Huang and Henry J Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. *Journal of the American medical informatics association*, 14(3):304–311.

- Aditya Khandelwal and Suraj Sawant. 2019. Negbert: A transfer learning approach for negation detection and scope resolution. *arXiv preprint arXiv:1911.04211*.
- Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyoungseob Park, and Priyadarshini Panda. 2020. Domain adaptation without source data. *arXiv preprint arXiv:2007.01524*.
- Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. Semeval 2018 task 6: Parsing time normalizations. In *proceedings of the 12th International Workshop on Semantic Evaluation*, pages 88–96.
- Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.
- Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. What’s in a domain? learning domain-robust text representations using adversarial training. *arXiv preprint arXiv:1805.06088*.
- Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Saeed Mehrabi, Anand Krishnan, Sunghwan Sohn, Alexandra M Roch, Heidi Schmidt, Joe Kesterson, Chris Beesley, Paul Dexter, C Max Schmidt, Hongfang Liu, et al. 2015. Deepen: A negation detection system for clinical text incorporating dependency relation into negex. *Journal of biomedical informatics*, 54:213–219.
- Amy Olex, Luke Maffey, Nicholas Morgan, and Bridget McInnes. 2018. Chrono at semeval-2018 task 6: a system for normalizing temporal expressions. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 97–101.
- David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.
- Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th international conference on World wide web*, pages 751–760.
- Barbara Plank, Anders Johannsen, and Anders Søgaard. 2014. Importance weighting and unsupervised domain adaptation of pos taggers: a negative result. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 968–973.
- Lior Rokach. 2010. Ensemble-based classifiers. *Artificial intelligence review*, 33(1):1–39.
- Antti Tarvainen and Harri Valpola. 2017. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *arXiv preprint arXiv:1703.01780*.
- Dongfang Xu, Egoitz Laparra, and Steven Bethard. 2019. Pre-trained contextualized character embeddings lead to major improvements in time normalization: A detailed analysis. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (\*SEM 2019)*, pages 68–74.
- Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 538–544.

Type	R(Baseline)	R(Ours)	Type	R(Baseline)	R(Ours)
Month-Of-Year	0.99	0.993	Between	0.812	0.79
Season-Of-Year	0.055	0.348	Two-Digit-Year	0.1	0
Year	0.988	0.988	Before	0.667	0.818
Number	0.88	0.855	After	0.915	0.872
Last	0.986	0.986	Frequency	0.6	0.6
Calendar-Interval	0.702	0.740	Next	0.839	0.839
Day-Of-Month	1.0	0.875	Intersection	0	0
This	0.528	0.537	Day-Of-Week	1.0	1.0
Period	0.816	0.827	NthFromStart	0	0
Modifier	0.826	0.855	Union	0	0
Part-Of-Day	1.0	1.0			

Table 6: Recall of each type of subtask B.

# SemEval-2021 Task 11: NLP CONTRIBUTIONGRAPH - Structuring Scholarly NLP Contributions for a Research Knowledge Graph

Jennifer D'Souza<sup>1</sup>, Sören Auer<sup>1</sup>, and Ted Pedersen<sup>2</sup>

<sup>1</sup>TIB Leibniz Information Centre for Science and Technology, Hannover, Germany

{jennifer.dsouza, auer}@tib.eu

<sup>2</sup> University of Minnesota, Duluth, USA

tpederse@d.umn.edu

## Abstract

There is currently a gap between the natural language expression of scholarly publications and their structured semantic content modeling to enable intelligent content search. With the volume of research growing exponentially every year, a search feature operating over semantically structured content is compelling. The SemEval-2021 Shared Task NLP CONTRIBUTIONGRAPH (a.k.a. ‘the NCG task’) tasks participants to develop automated systems that structure contributions from NLP scholarly articles in the English language. Being the first-of-its-kind in the SemEval series, the task released structured data from NLP scholarly articles at three levels of information granularity, i.e. at sentence-level, phrase-level, and phrases organized as triples toward Knowledge Graph (KG) building. The sentence-level annotations comprised the few sentences about the article’s contribution. The phrase-level annotations were scientific term and predicate phrases from the contribution sentences. Finally, the triples constituted the research overview KG. For the Shared Task, participating systems were then expected to automatically classify contribution sentences, extract scientific terms and relations from the sentences, and organize them as KG triples.

Overall, the task drew a strong participation demographic of seven teams and 27 participants. The best end-to-end task system classified contribution sentences at 57.27% F1, phrases at 46.41% F1, and triples at 22.28% F1. While the absolute performance to generate triples remains low, in the conclusion of this article, the difficulty of producing such data and as a consequence of modeling it is highlighted.

## 1 Introduction

Traditional search models over scholarly communication are now changing toward Knowledge Graph (KG) models operating on structured fine-grained

scholarly content offering enhanced contextual search results. Several initiatives exist to this end: Google Scholar, Web of Science (Birkle et al., 2020), Microsoft Academic Graph (Wang et al., 2020), OpenAIRE Research Graph (Manghi et al., 2019), Open Research Knowledge Graph (Auer, 2018), Semantic Scholar (Fricke, 2018) to name just a few. These KG models differ in their content, their level of detail, etc., as they represent diverse aspects of scholarly communication.

Text, of course, is of seminal importance to Science. It is as important as experimentation itself; unpublished research lacks validity. Seen in another angle, it is hard to imagine a medium other than discourse that can convey a comprehensive picture of the scholarly investigation. For the wider research audience, it is interesting to read the full “stories” of Science.

Nonetheless, since scientific literature is growing at a rapid rate (Johnson et al., 2018) and researchers today are faced with this publications deluge (Landhuis, 2016), it is increasingly tedious, if not practically impossible to keep up with the research progress even within one’s own narrow discipline. In this regard, among the existing scholarly knowledge structuring initiatives, the Open Research Knowledge Graph (ORKG) (Auer et al., 2020) is posited as a solution to the problem of keeping track of research progress minus the cognitive overload that reading dozens of full papers impose. It aims to build a comprehensive KG that publishes the research contributions of scholarly publications per paper, where the contributions are interconnected via the graph even across papers. The ORKG digital library (DL) framework can be accessed here <https://www.orkg.org>.

Motivated by the availability of a next-generation DL, we present the SemEval-2021 NLP-CONTRIBUTIONGRAPH (NCG) Shared Task as a step in the easier knowledge acquisition of contri-



butions for researchers - *the automated structuring of the unstructured article contributions*. To this end, via the NCG task, we have formalized the building of such a scholarly contributions-focused graph over NLP scholarly articles as an automated task. In the subsequent paper, we detail our task in terms of its resources, organization, participants, and evaluations.

## 2 Data

The NCG Shared Task comprised a dataset of NLP scholarly articles annotated for their contributions. The contributions were structured to be integrable within KG infrastructures such as the ORKG (Jaradeh et al., 2019) that capture research overviews. The contributions were annotated in three different information granularities, i.e. (1) *Contribution sentences*: a set of sentences about the article’s contribution; (2) *Scientific terms and relations*: a set of terms and relational predicates in the contribution sentences; and (3) *Triples*: semantic statements that pair the terms with a predicate, modeled toward subject-predicate-object RDF statements for KG building. This latter set of annotations formed the actual graph. Inspired after article sections, the Triples were organized under three (mandatory) or more of 12 total information units (IUs), viz. RESEARCHPROBLEM, APPROACH, MODEL, CODE, DATASET, EXPERIMENTALSETUP, HYPERPARAMETERS, BASELINES, RESULTS, TASKS, EXPERIMENTS, and ABLATIONANALYSIS.

### 2.1 Data Annotation Scheme

A trial annotation stage preceded the annotation of the Shared Task dataset. In this stage, an annotation scheme was prescribed. This involved specifying the annotation data granularities and the 12 IUs for organizing the triples. Observations were also obtained about the position in the articles where the authors generally stated the contribution. The trial annotations were conducted in two steps: a pilot annotation step (D’Souza and Auer, 2020) followed by an adjudication step (D’Souza and Auer, 2021). The resulting scheme itself was called the NLPCONTRIBUTIONGRAPH (NCG) scheme.

For the trial stage, a relatively small dataset of 50 articles uniformly distributed across five NLP tasks, i.e. machine translation, named entity recognition, question answering, relation classification, and text classification, were selected.

Overall, after the pilot annotation task the follow-

ing core question was answered. Could a scheme be defined such that it would encompass all annotation decisions of the task? In reality, it was found that the scheme could only define high-level annotation decisions such as: where in the article could the contribution information generally be found? E.g., the title, the abstract, a few lines in the Introduction, the first few lines of the Results section. This still entailed making subjective decisions such as if the model is not described in the Introduction then the first few lines of the model description section would need to be annotated. The scheme also specified the 12 IUs for organizing the structured triples. The choice of the specific IU for organizing the triples was based on the closest section title.

After the two-step trial annotation stage, the intra-annotation agreement between the pilot and adjudication steps, in terms of F1, was 67.92% for sentences, 41.82% for phrases, and 22.31% for triple statements indicating that with increased granularity of the information, the annotation adjudication was greater (2021).

The trial annotations were made by a postdoctoral researcher in Computational Linguistics. The same experienced annotator also annotated the full dataset. Next, we explain the NCG data with a focus on the KG and then offer two supporting examples as illustrations of the data.

### 2.2 Understanding our Knowledge Graph

The NCG KG used two levels of knowledge systematization: 1) At the root, it defined a dummy node called CONTRIBUTION. And following the root node, 2) it defined the 12 nodes introduced earlier and generically referred to as Information Units or IUs. Each scholarly article’s annotated contribution triple statements were organized under three (mandatory) or more of these IU nodes, depending on whether they applied to the article. Next, we provide details about each IU.

**RESEARCHPROBLEM** The research challenge addressed by a contribution. In other words, a focus of the research investigation or the issue for which a research solution was proposed.

**APPROACH or MODEL** The contribution of the paper as the solution proposed for the research problem. This unit was called APPROACH when the solution was proposed as an abstraction, and was called MODEL if the solution was proposed in practical implementation terms. Further, in case the solution was not referred to as approach or

model in the article, the reference was normalized as either APPROACH or MODEL. E.g., references like “method” or “application” were normalized as APPROACH; on the other hand, references like “system” or “architecture,” were normalized to MODEL. This unit captured only proposed system highlights.

**CODE** The contribution resource; the link to the software on an open-source hosting platform such as Gitlab or Github or on the author’s website.

**DATASET** Like CODE, this a contributed resource in the form of a dataset.

**EXPERIMENTALSETUP or HYPERPARAMETERS** Details about the platform including both hardware (e.g., GPU) and software (e.g., Tensorflow library) for implementing the machine learning solution; and of variables, that determine the network structure (e.g., number of hidden units) and how the network is trained (e.g., learning rate), for tuning the software to the task objective. It was called EXPERIMENTALSETUP only when hardware details were provided.

**BASELINES** The systems that a proposed APPROACH or MODEL were compared with.

**RESULTS** The main findings or outcomes reported in an article for the RESEARCHPROBLEM.

**TASKS** The APPROACH or MODEL, particularly in multi-task settings, are tested on more than one task, in which case, this unit was defined to capture all the experimental tasks. Unlike the earlier units, the TASKS IU was a container for more than one of the earlier mentioned IUs. Specifically, each task listed in TASKS could include one or more of the EXPERIMENTALSETUP, HYPERPARAMETERS, and RESULTS as sub-information units.

Furthermore, since it is common in NLP for tasks to be defined over datasets, experimental tasks are often synonymous with the experimental datasets, therefore this unit was also applied in articles where the datasets were explicitly listed instead of the task names.

**EXPERIMENTS** The second container information unit, like TASKS, defined to include one or more of the previous discussed units as sub-information units. This unit encapsulated several TASKS themselves and consequently, the units that TASKS encapsulated, i.e. EXPERIMENTALSETUP and RESULTS, or a combination of APPROACH, EXPERIMENTALSETUP and RESULTS.

**ABLATIONANALYSIS** A form of RESULTS that describes the performance of components in an APPROACH or MODEL.

### 2.3 Data Examples

Below, we show two examples of two different IUs, viz. RESEARCHPROBLEM and MODEL, respectively, as illustrations of our data.

```
{
  "has research problem" : [
    [ "Statistical Machine Translation",
      { "from sentence": "Learning Phrase
        Representations using RNN Encoder -
        Decoder for Statistical Machine
        Translation"} ],
    [ "SMT", "phrase - based SMT", { "from
      sentence" : "Along this line of
        research on using neural networks
        for SMT , this paper focuses on a
        novel neural network architecture
        that can be used as apart of the
        conventional phrase - based SMT
        system ."} ]
  ]
}
```

Figure 1: Annotated data in JSON format for the RESEARCHPROBLEM Information Unit for the paper “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.”

**Example 1** In this example, the RESEARCHPROBLEM IU is modeled for the following reference paper: *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation* (Cho et al., 2014). We show two formats of our data: the JSON format (see Fig. 1) with all three annotated information granularities; and the triples format (see Table 1) showing only the annotated data for a KG. In the JSON data, the dummy root node CONTRIBUTION is left unspecified, however, it is specified in the triples. For this data, three phrases that named the research problem were annotated. The phrases were attached to the dummy root node by the predicate “has research problem.” Further, in the JSON data, following the predicate “from sentence,” the selected contribution sentences are listed.

**Example 2** In this example, a subpart of the MODEL IU is annotated for the following reference paper: *Convolutional Neural Network Architectures for Matching Natural Language Sen-*

(Contribution, has, Statistical Machine Translation)
(Contribution, has, SMT)
(Contribution, has, Phrase - Based SMT)

Table 1: Annotated RESEARCHPROBLEM Information Unit contribution data as triples. This data is obtained from the JSON data shown in Fig 1.

tences (Hu et al., 2014). See Fig. 2 for the JSON format and Table 2 for the triples data.

```

{
  "has" : {
    "Model" : {
      "propose" : {
        "deep neural network models" : {
          "adapt" : {
            "convolutional strategy" : {
              "to" : "natural language"
            }
          }
        }
      },
      "from sentence" : "Towards this end , we propose deep neural network models , which adapt the convolutional strategy ( proven successful on image and speech ) to natural language ."
    }
  }
}

```

Figure 2: Annotated data in JSON format for the MODEL Information Unit for the paper “Convolutional Neural Network Architectures for Matching Natural Language Sentences.”

(Contribution, has, Statistical Machine Translation)
(Model, propose, deep neural network models)
(deep neural network models, adapt, convolutional strategy)
(convolutional strategy, to, natural language)

Table 2: Annotated MODEL Information Unit contribution data as triples. This data is obtained from the JSON data illustrated in Fig. 2.

## 2.4 Data Statistics

Overall, the NCG Shared Task dataset had 50 articles in the trial data, 237 articles in the training data, and 155 articles in the test data. The trial data articles uniformly spanned five tasks, the training data spanned 24 tasks, and the test data spanned 10

tasks. For the Shared Task itself, participants were encouraged to merge the trial and training datasets. Thus, the overall training data had 287 articles representing 29 unique tasks. The training and test tasks were mutually exclusive except for one, i.e. ‘natural language inference.’ Table 3 shows further detailed statistics of the NCG dataset in terms of each of the annotated information granularities.

Our full dataset is publicly released online (D’Souza et al., 2021).

## 3 Task Description

Our comprehensive NCG Shared Task formalism was as follows. Given a scholarly article  $A$  in plaintext format, the goal was to extract (1) a set of contribution sentences  $C_{sent} = \{C_{sent_1}, \dots, C_{sent_N}\}$ , (2) a set of scientific knowledge terms and predicates from  $C_{sent}$  referred to as entities  $E = \{e_1, \dots, e_N\}$ , and (3) to organize the entities  $E$  as a set of (subject,predicate,object) triple statements  $T = \{t_1, \dots, t_N\}$  toward KG building organized under three or more of the 12 total IUs.

**Task Evaluation Phases.** The task comprised three evaluation phases, thereby enabling detailed system evaluations.

*Evaluation Phase 1: End-to-end Pipeline.* In this phase, systems were tested for the comprehensive end-to-end KG building task described in the formalism above. Given a test set of articles  $A$  in plaintext format, the participating systems were expected to return: (1) a set of contribution sentences  $C_{sent}$ , (2) a set of scientific knowledge terms and predicates from  $C_{sent}$ , i.e. entities  $E$ , and (3) the entities in  $E$  organized in a set of triple statements  $T$  toward KG building. System outputs were evaluated for the three aspects and overall.

*Evaluation Phase 2, Part 1: Phrases and Triples.* In this phase, systems were tested only for their capacity to extract phrases and organize them as triples. Given a test set of articles  $A$  in plain-text format and contribution sentences  $C_{sent}$  from each article, each system was expected to return: (1) the entities  $E$ , and (2) the set of triple statements  $T$ .

*Evaluation Phase 2, Part 2: Triples.* In this phase, systems were tested only for the triples formation task. Thus, given gold entities  $E$  for the set of  $C_{sent}$ , systems were expected to form triple statements  $T$ .

Dataset	<i>info-units</i>	<i>sentences</i>	<i>entities</i>	<i>total triples</i>	<i>total unique triples</i>	<i>subject</i>	<i>predicate</i>	<i>object</i>
TRIAL	217	1,029	4,777	2,924	2,782	1,427	1,181	2,512
TRAIN	1,050	5,064	30,485	18,679	17,356	8,173	4,538	13,335
TEST	642	2,720	16,435	10,623	10,002	4,951	2,447	8,282

Table 3: NLPCONTRIBUTIONGRAPH Shared Task 2021 Overall Corpus Statistics

## 4 Task Setup

**Online Competition** We used the CodaLab platform for running the competition online <https://competitions.codalab.org/competitions/25680>. For the convenience of the participants, the task was divided into four phases. In the Practice phase, which began on Aug 16, 2020, we released the participant kit that included the full training dataset along with the Python code of the official scoring program <https://github.com/ngc-task/scoring-program>. In the Evaluation phases that lasted from Jan 10 till Feb 1, 2021, we provided the participants with masked versions of the test set based on the current evaluation phase. The test set annotations in each phase were uploaded to CodaLab and were not available to the participants. To obtain results, the participants were expected to upload their system outputs to CodaLab where they were automatically evaluated by our script and reference data stored on the platform. In each evaluation phase, teams were restricted to make only 10 submissions and only one result, i.e. the top-scoring result, was shown on the leaderboard.

Before the task began, our participants were onboarded via our task website <https://ngc-task.github.io/>. Further, participants were encouraged to discuss their task-related questions via our task Google groups page at <https://groups.google.com/forum/#!forum/ngc-task-semeval-2021>.

**The NCG Data Collection of Articles** Our base collection of scholarly articles was downloaded from the publicly available leaderboard of tasks in AI called <https://paperswithcode.com/>. While *paperswithcode* predominantly represents the NLP and Computer Vision research fields in AI, we restricted ourselves just to its NLP papers. From their overall collection of articles, the tasks and articles in our final data were randomly selected. The raw articles’ pdfs needed to undergo a two-step preprocessing before the annotation task. 1) For pdf-to-text conversion, the GROBID parser (GRO,

2008–2020) was applied; following which, 2) for plaintext pre-processing in terms of tokenization and sentence splitting, the Stanza toolkit (Qi et al., 2020) was used. The resulting pre-processed articles could then be annotated in plaintext format. Note, our data consists of articles in English.

**Evaluation Metrics** The NCG Task participating team systems were evaluated for classifying contribution sentences, extracting scientific terms and relations, and extracting triples (see specific details in Section 3). The results from the three evaluations parts were also cumulatively averaged as a single score to rank the teams. Finally, for the evaluations, the standard precision, recall, and F1-score metrics were leveraged.

This completes our discussion of the NCG task in terms of its dataset definition and overall organization description. In the remainder of the paper, we shift our focus to the participating teams. Specifically, we describe the participating systems and examine their results for the NCG task.

## 5 Participating System Descriptions

The NCG Shared Task received public entries from 7 participating teams in all. In this section, we briefly describe the teams’ systems in terms of the three parts of the NCG task, i.e. contribution sentence classification, scientific terms and relations extraction, and triples extraction.

### 5.1 Contribution Sentence Classification

To identify the contribution sentences from articles, systems adopted one of two strategies: a binary classification objective, or a multi-class classification objective. In the first strategy, sentences were either classified as contribution sentences or not. In the second strategy, sentences were classified in a 13-class classification task as one of the 12 IUs or as a non-contribution sentence. Next, we describe these strategies. Note, the asterisk superscripts against team names, where present, correspond to \*\*\* 3rd best, \*\* 2nd best, and \* 1st best systems in the Shared Task, respectively.

**Binary Classifiers** Team YNU-HPCC (Ma et al., 2021) employed BERT as a *binary classifier* to classify the contribution sentences. Team INNOVATORS (Arora et al., 2021) also employed a BERT-based *binary classifier* wherein each instance was a set of 10 sentences with additional sentences as context features to the model. Team KnowGraph@IITK\*\*\* (Shailabh et al., 2021) used the standard SciBERT + BiLSTM architecture (Beltagy et al., 2019) as a *binary sentence classifier*. Team UIUC.BioNLP\* (Liu et al., 2021) employed BERT-based *binary sentence classifier* with features that handled sentence characteristics w.r.t. their context in the article - specifically, its closest preceding topmost and innermost section headers and its position in the article.

**Multi-class Classifiers** Team DULUTH (Martin and Pedersen, 2021) framed a *13-class multi-class classification* task. They employed deBERTa (He et al., 2020) as their classifier. Team ECNUICA (Lin et al., 2021) employed three pre-trained transformer models, viz. RoBERTa (Liu et al., 2019), SciBERT (Beltagy et al., 2019), and BERT (Devlin et al., 2019) as an ensemble classifier. They formulated a multi-class classification task as well. The features to BERT models are the original sentence, contextual information as previous and next sentence to the original sentence, and a sub-title of the paragraph with the separator token ([SEP]) in between. Team ITNLP\*\* (Zhang et al., 2021) employed a BERT-based multi-class classifier that leveraged sentence context and the paragraph heading as additional features.

These binary and multi-class sentence classifiers, were also adapted to our following dataset characteristics.

### 5.1.1 Contribution sentences data imbalance

Characteristically, of all the sentences in training data scholarly articles, only 10% were annotated as contribution sentences. Thus, our dataset presented an imbalanced classification task.

Teams YNU-HPCC, DULUTH, KnowGraph@IITK\*\*\* and UIUC.BioNLP\* trained their classifiers on the given data. While INNOVATORS and ITNLP\*\* downsampled the non-contribution sentences. INNOVATORS established a threshold based on cumulative contributing sentence bigram scores as a filter; ITNLP fixed the ratio of positive to negative samples as an integer and tuned the value.

### 5.1.2 Differing tasks coverage between the training and the test datasets

Since only one task was in common between the training and the test datasets, this meant that systems trained only on the training data would be applied on articles from nine new tasks as test data. To this end, Team ECNUICA hypothesized that if the classifier could *see*, i.e. somehow be trained on, the test data tasks, its performance could be boosted. They, thus, adopted the strategy of re-training their classification ensemble with silver-labeled test data instances. This followed the standard setup of training the classifier on the actual training data, applying it to the test data, and incrementally retraining the classifier leveraging the few confidently classified test instances. The instances were marked as silver training data only when all three ensemble classifiers predicted the same class.

### 5.2 Scientific Terms and Relations Extraction

After identifying the contribution sentences, systems then had to extract their scientific terms and relational predicates.

**Sequence Labeling Systems** Majority, i.e. six, of the seven participating systems adopted a sequence labeling approach.

1. Team YNU-HPCC used a pre-trained BERT model for sequence labeling of each token, obtaining embeddings for each token in the sequence, with softmax and argmax top layers which were shared across all tokens.
2. Team DULUTH trained a feature-based maximum-entropy Markov model (MEMM) to predict scientific terms in the contribution sentences.
3. Team ECNUICA extracted entities using RoBERTa (Liu et al., 2019) with a CRF layer and a BIO sequence labeling scheme. The input sequences to RoBERTa are modified with sub-title information.
4. Team KnowGraph@IITK\*\*\* extracted phrases in the sentence by adding BiLSTM layers to the SciBERT + CRF model as a sequence labeler. To mark phrase boundaries, they used the BILUO scheme.
5. Team ITNLP\*\* employed the standard BERT-based model, however, in a sequence labeling setting. They trained ten different models

by 10-fold cross-validation and used a voting count threshold scheme to extract the final set of entities.

6. Team UIUC\_BioNLP\* used a BERT-CRF model for phrase extraction and type classification (Souza et al., 2019). They employed the BIO scheme to distinguish the scientific terms vs. predicate phrases.

**Rule-based System** Team INNOVATORS leveraged an unsupervised rule-based approach for phrase extraction. Using spaCy (Honnibal et al., 2020), they obtained dependency parses for each sentence. They then implemented a set of dependency tree node traversal heuristics for phrase extraction based on the dependency parses.

### 5.3 Triples Extraction

1. Team YNU-HPCC first classified the scientific terms in subject, predicate, and object roles using three binary BERT classifiers. These triples from each contribution sentence were then organized as the 12 IUs leveraging a 12-class *contribution* sentence classifier. This team, however, did not participate in the end-to-end evaluation task.
2. Team DULUTH applied Stanford Core NLP’s dependency parser (Chen and Manning, 2014) to generate a dependency parse for each contribution sentence. They used the dependency parse structures to assign subject, relation, and object phrase roles to the extracted scientific terms. These were then organized as triples per IU obtained by their 13-class sentence classifier. *The overall end-to-end pipeline system score achieved by this system is 28.38%.*
3. Team INNOVATORS implemented a set of rules based on the dependency parses to form triples from the extracted scientific terms. They used a CNN-based architecture for classifying the *contribution* sentences as the 12 IUs. *Their end-to-end score was 32.05%.*
4. Team ECNUICA approached the triples formation task in two steps: i) they formed triple candidates based on the scientific term sequence order in the sentence. Additionally, they employed a set of predefined predicates when the predicates were not directly found in the sentence. ii) They then employed a SciBERT-based binary classifier to classify the triples as true or false candidates. *Their overall end-to-end system score was 33.35%.*
5. Team KnowGraph@IITK\*\*\* addressed the RESEARCHPROBLEM, CODE, BASELINES and ABLATIONANALYSIS IUs by a heuristics-based approach. For the remaining eight IUs triples, they followed a 3-step approach: i) identify predicates from the scientific terms using a binary SciBERT+BiLSTM classifier; and ii) formed triples by arranging the terms and predicates in exact order as they appear in the original sentence; and iii) employ an 8-class SciBERT + BiLSTM classifier to classify the triples. *Their overall end-to-end system score was 37.83%.*
6. Team ITNLP\*\* extracted triples as follows: i) they formed all possible triples candidates from the classified scientific terms; and ii) employed a binary BERT classifier for true or false candidates. Prior to BERT classification, they perform the negative candidate triples downsampling as follows: by artificially generating them using random replacement (RR) of one of the arguments of the true triples with a false argument; and by random selection (RS) of triples where no argument is a valid pair of another. Additionally, each of their system components obtained boosted performances with the Friendly Adversarial Training strategy (Zhang et al., 2020). *Their overall end-to-end system score was 47.03%.*
7. Team UIUC\_BioNLP\* categorized the triples into six types based on our dataset characteristics. Four of the six types were: structuring intra-sentence information; linking sentence information to IU; linking IU to the root node; and structuring inter-sentence information. The first two of the four broad types were further subdivided into two based on whether the predicate was found in the sentence or was the term “has.” Each of the six types were addressed by a specifically trained BERT classifier. *They obtained an overall end-to-end system score of 38.28% within the task deadline and 49.72% a day later after fixing phrase component offset errors.*<sup>1</sup>

<sup>1</sup>Per the task timeline, i.e. within the Phase 1 end-to-end system evaluation, the team achieved 38.28% F1 within the

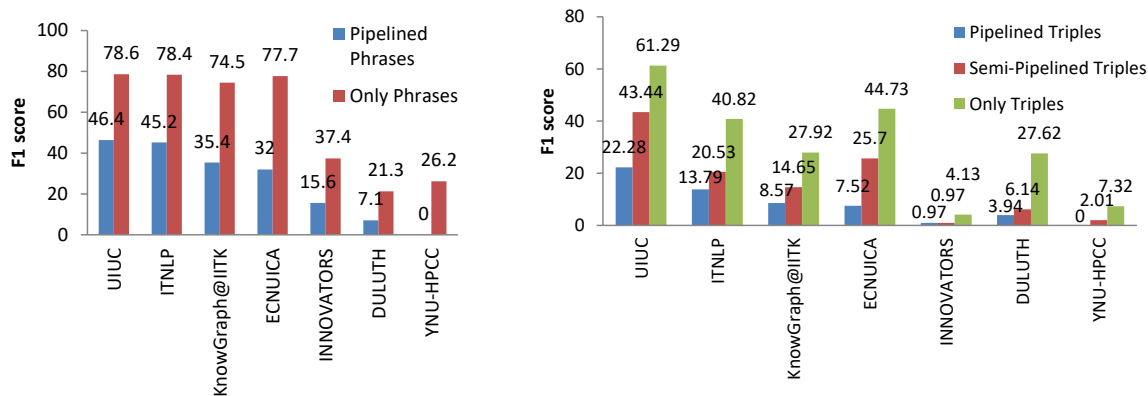
	1	2.1	2.2
ITNLP <a href="https://github.com/itnlp606/nlpcb-graph">https://github.com/itnlp606/nlpcb-graph</a>	<b>47.03</b>	68.63	79.31
UIUC_BioNLP <a href="https://github.com/Liu-Hy/nlp-contrib-graph">https://github.com/Liu-Hy/nlp-contrib-graph</a>	38.28**	<b>76.12</b>	<b>85.94</b>
KnowGraph@IITK <a href="https://github.com/sshailabh/SemEval-2021-Task-11">https://github.com/sshailabh/SemEval-2021-Task-11</a>	37.83	63.18	76.0
ECNUICA	33.35	71.13	81.45
INNOVATORS <a href="https://github.com/HardikArora17/SemEval-2021-INNOVATORS">https://github.com/HardikArora17/SemEval-2021-INNOVATORS</a>	32.05	52.52	59.71
DULUTH <a href="https://github.com/anmartin94/DuluthSemEval2021Task11">https://github.com/anmartin94/DuluthSemEval2021Task11</a>	28.38	49.21	45.62
YNU-HPCC <a href="https://github.com/maxinge8698/SemEval2021-Task11">https://github.com/maxinge8698/SemEval2021-Task11</a>		75.79	65.41

Table 4: The seven NLPCONTRIBUTORGRAPH participating teams with their averaged F1 scores over individual subtasks per evaluation phase. Column “1” - Evaluation Phase 1: End-to-end Pipeline F1; Column “2.1” - Evaluation Phase 2, Part 1: Phrases and Triples F1; and Column “2.2” - Evaluation Phase 2, Part 2: Triples Extraction F1.

\*\*system submission had error in phrase offsets for task submission; actual task performance was 49.72 F1.

Model	Sentences			Phrases			Information Units			Triples		
	F1	P	R	F1	P	R	F1	P	R	F1	P	R
UIUC_BioNLP	57.27	<b>53.61</b>	61.46	46.41	<b>42.69</b>	50.83	72.93	66.67	80.49	22.28	<b>22.3</b>	<b>22.26</b>
ITNLP	56.19	51.74	61.46	45.22	41.6	49.55	72.93	66.67	80.49	13.79	13.39	14.23
KnowGraph@IITK	46.8	39.69	57.01	35.4	28.99	45.44	60.54	44.13	<b>96.34</b>	8.57	6.53	12.45
ECNUICA	39.78	26.21	<b>82.48</b>	32.03	20.73	<b>70.37</b>	54.05	42.86	73.17	6.78	4.28	16.29
INNOVATORS	39.87	39.32	40.45	15.63	13.27	19.01	71.72	<b>82.54</b>	63.41	0.97	14.29	0.5
DULUTH	38.1	44.83	33.12	7.08	13.07	4.86	64.41	60.0	69.51	3.94	9.2	2.51

Table 5: Evaluation Phase 1: End-to-end Pipeline Results



(a) Evaluation Phase 1: End-to-end Pipeline Pipelined Phrases (blue bars) and Evaluation Phase 2, Part 1: Phrases and Triples Only Phrases (red bars) results (b) Evaluation Phase 1: End-to-end Pipeline Pipelined Triples (blue bars); Evaluation Phase 2, Part 1: Phrases and Triples Semi-Pipelined Triples (red bars); and Evaluation Phase 2, Part 2: Triples Only Triples (green bars) results

Figure 3: (a) Phrases and (b) Triples extraction results

## 6 Shared Task Results

In this section, we present the results of the seven participating teams’ systems.

The results in Table 4 show the cumulative scores of the participating teams in each of the three evaluation phases in our Shared Task. We refer the reader to Section 3 for a detailed description of the three evaluation phases. In each phase, Teams were officially ranked by these scores. Next, we examine the scores by the individual extraction

task deadline due to an error in their submission offsets for phrases. Thus, they are officially 2nd after the ITNLP team within the Shared Task timeline for Phase 1.

tasks that constituted building the NLPCONTRIBUTORGRAPH per article.

### 6.1 Contribution Sentences Classification

As a first step toward building the NLPCONTRIBUTORGRAPH, systems were evaluated for identifying contribution sentences. This was done only in the Evaluation Phase 1 of the Shared Task, i.e. the phase that tested the end-to-end systems. These results are shown in Table 5 under column “Sentences.” This subtask attained a high score of 57%. The top two teams, i.e. UIUC\_BioNLP\* and ITNLP\*\*, differed by only 1 point. Comparing these performances to a baseline, a default system

would return all titles as candidate contribution sentences. This results in a score of 10.78% F1 at 90% precision and 5.7% recall. In contrast to the 1 sentence per article result in the default computation, our actual data averages at 17 sentences per article. Thus the default score was computed on a significantly underestimated data sample as also reflected by its low recall. Nevertheless, the top systems significantly outperform this default score with both systems averaging at 20 sentences per article. The least score was also significantly better than the default at 38.1% F1 at an average of 12 sentences per article.

With F1 less than 60%, the task shows itself challenging. Some teams ascribed this to the dataset characteristic that contribution sentences constituted only a minority of the sentences in the article (<10%) and thus, overall, presented imbalanced data. To address this they downsampled the data. However, from the two participant systems that used a downsampling strategy, it could not be conclusively verified as an effective strategy since these systems performed on opposite ends of the performance spectrum. On the other hand, *incorporating the closest section header and sentence position as features in the BERT model showed itself an effective and reliable strategy for sentence classification*. This modeled the dataset better since the sentences were annotated from a few sections and the sentences were usually close to the section header. The system UIUC\_BioNLP\* that incorporated such features outperformed all other systems including the ones with the downsampling strategy, i.e. ITNLP\*\* and INNOVATORS.

Finally, how did bootstrapping the test data as silver-labeled data impact model performance? Team ECNUICA that adopted this strategy did not obtain a balanced harmonic mean between their precision and recall achieving the highest recall among all teams of 82.48% and the lowest precision of 26.21%. Thus this strategy did not show itself too effective and reliable.

## 6.2 Scientific Terms and Relations Extraction

These results are shown in Table 5 under column “Phrases” for the end-to-end systems. The highest F1 obtained on this task was 46.41%. However, this score was impacted by the pipeline setup such that the low performance in sentence classification impacted the performance in this stage. We conducted a separate evaluation phase to control for this as-

pect. In other words, we examined how would the systems perform only on extracting terms and relations given gold contribution sentences? These results are shown in Figure 3 (a). In fact, the bar chart offers a perspective on the significant differences in system performances when applied on automatically extracted sentences versus gold data. The systems showed the same performance ranking order in both settings. This is a somewhat expected result since none of the systems implemented any specific noisy sentence handling strategy in which case performance differences may have risen. In conclusion, the best result was 46.4% F1 in the end-to-end setting and was 78.6% F1 when given gold sentences.

Notably, the pipeline systems were 10 points lower for extracting phrases than for sentences.

## 6.3 Triples Extraction

The final extraction task to build the NCG per article was to form triples from the extracted terms and relations. These results for the pipeline systems are shown in Table 5 under column “Triples.” The best performance was 22.28% F1 and the 2nd best was significantly lower at 13.79% F1. To evaluate system performances purely for extracting triples, thereby cancelling out the effect of the pipeline setup, additional evaluations were conducted wherein gold data were incrementally made available to the system. These results are shown in Figure 3 (b). Given only the gold sentences, the best team attained 43.44% F1; given gold terms and relations in addition, they achieved 61.29% F1. A score of 61.29% F1 is a strong performance on a still fairly difficult task given the annotation decision subjectivity that may have crept into the data thereby producing considerable variations in annotation patterns. This is discussed in Section 7.

### Identifying only the Information Unit Labels

We conducted a meta-evaluation for identifying the set of IU labels per article. These results are shown in Table 5 under column “Information Units.” The top two teams were tied at 72.93% F1 with the second best score at 60.54% F1. Like sentence classification, a default system could be implemented for this task as one that output just the three mandatory IUs, i.e. RESEARCHPROBLEM, MODEL, and RESULTS for all articles. The scores from this default system were 69.01% F1, 81.67% precision, and 59.76% recall. It is 9 points better than the 2nd best. When given gold sentences, systems could



be evaluated for identifying just the IUs since the classification were dependent on the underlying sentences. These results are shown in Fig. 4.

A notable exception in the results is that the IU classification score by Team INNOVATORS remained unchanged regardless of pipelined or gold sentences as input. This is because their downsampling heuristic once designed did not rely on the underlying data when filtering. It is likely that the new gold sentences information was not used at all.

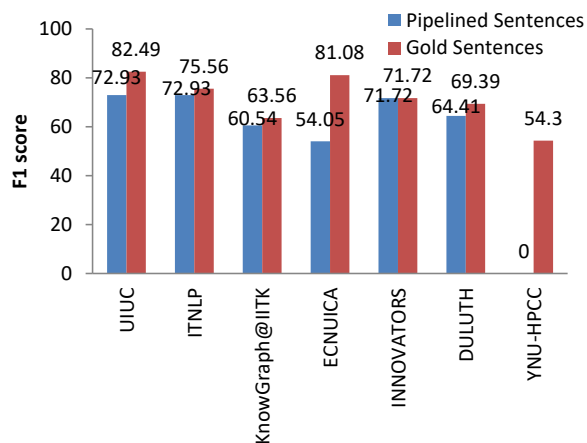


Figure 4: Information Unit identification results in *Evaluation Phase 1: End-to-end Pipeline with Pipelined Sentences* (blue bars) and *Evaluation Phase 2, Part 1 and Part 2 with Gold Sentences* (red bars)

## 7 Discussion

Finally, we conclude our Shared Task paper with a discussion on the perceived limitations of our dataset that can potentially be addressed in future work. Thereby, a new dataset will present new opportunities to evaluate systems on this novel task.

**Single Annotator Annotations** The NCG Shared Task dataset was annotated by a single annotator. Further, the design of the annotation scheme was supported by only an intra-annotator consensus agreement score for that annotator. Since this work is the first-of-its-kind in proposing an initial scheme, and given the complex nature of this annotation task with the need to design a model within a realistic timeframe, our annotation procedure is well-suited.

However, as discussed in our related work (D’Souza and Auer, 2021), in the next stage, we advocate for a blind, multi-stage, and multi-annotator annotation process for the NCG scheme, recognizing it as a potentially better annotation model. We find that such a process

while incorporating multiple worldviews could better address annotation inconsistencies that may have crept in in our current dataset.

**Non-uniform Distribution of Articles** As discussed earlier, our combined training dataset had 29 tasks and the test data had 10 tasks. However, these tasks did not have a uniform distribution of articles in our data. In the training data, the number of articles per task ranged from a maximum of 101 in one task, i.e. “natural language inference,” to a minimum of one article in seven tasks - 58.62% of the training data tasks had less than 5 articles. The test dataset, on the other hand, followed a more uniform distribution than the training data ranging from a maximum of 32 articles to a minimum of seven articles at an average of 15.5% articles per task. While our training dataset had over 200 articles, it may not have been sufficiently representative to learn uniform patterns. Thus in a new version of the dataset, a more uniform representation of the tasks will be attempted.

## 8 Conclusions

We have detailed the NLPCONTRIBUTIONGRAPH Shared Task that entailed structuring research contributions in NLP articles as structured KGs. This task is the first-of-its-kind to be organized in the SemEval series. It attracted a strong participation demographic of 27 participants and seven teams - BERT transformer models were a popular choice among the participant systems in two different capacities, i.e. as classifiers or sequence labelers. Our task also saw the use of traditional parsers such a dependency syntax parsing technology. Further, some systems leveraged a hybrid approach including a combination of heuristics and machine learning. While the end-to-end task performance was low showing the task considerably challenging, each individual subtask toward obtaining an NCG, i.e. contribution sentence classification, scientific terms and relations extraction, and triples formation, demonstrated high performances in the subtask-only evaluation setting, i.e. when given gold data from the previous stage. The best system adopted a hybrid approach which seemed the most effective strategy for building the NCG.

The NCG dataset is publicly available (D’Souza et al., 2021) and a KG overview of a structured form of our paper is here <https://www.orkg.org/orkg/comparison/R74774>.

## Acknowledgments

We thank the anonymous reviewers for their comments and suggestions. This work was co-funded by the European Research Council for the project ScienceGRAPH (Grant agreement ID: 819536) and by the TIB Leibniz Information Centre for Science and Technology.

## References

- 2008–2020. GROBID. <https://github.com/kermitt2/grobid>.
- Hardik Arora, Sandeep Kumar, Tirthankar Ghosal, Suraj Patwal, and Phil Gooch. 2021. INNOVATORS at SemEval-2021 Task 11: A Dependency Parsing and BERT-based model for Extracting Contribution Knowledge from Scientific Papers. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.
- Sören Auer, Allard Oelen, Muhammad Haris, Markus Stocker, Jennifer D’Souza, Kheir Eddine Farfar, Lars Vogt, Manuel Prinz, Vitalis Wiens, and Mohamad Yaser Jaradeh. 2020. Improving Access to Scientific Literature with Knowledge Graphs. *Bibliothek Forschung und Praxis*, 44(3):516–529.
- Sören Auer. 2018. Towards an Open Research Knowledge Graph.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: Pretrained Language Model for Scientific Text. In *EMNLP*.
- Caroline Birkle, David A Pendlebury, Joshua Schnell, and Jonathan Adams. 2020. Web of science as a data source for research on scientific and scholarly activity. *Quantitative Science Studies*, 1(1):363–376.
- Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Jennifer D’Souza and Sören Auer. 2020. NLPContributions: An Annotation Scheme for Machine Reading of Scholarly Contributions in Natural Language Processing Literature. In *Proceedings of the 1st Workshop on Extraction and Evaluation of Knowledge Entities from Scientific Documents (EEKE 2020) co-located with the ACM/IEEE Joint Conference on Digital Libraries in 2020 (JCDL 2020)*, pages 16–27.
- Jennifer D’Souza and Sören Auer. 2021. Sentence, Phrase, and Triple Annotations to Build a Knowledge Graph of Natural Language Processing Contributions—A Trial Dataset. *Journal of Data and Information Science*, 0(0):–.
- Jennifer D’Souza, Sören Auer, and Ted Pederson. 2021. SemEval-2021 Task 11: NLPContributionGraph - Structuring Scholarly NLP Contributions for a Research Knowledge Graph.
- Suzanne Fricke. 2018. Semantic scholar. *Journal of the Medical Library Association: JMLA*, 106(1):145.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. *arXiv preprint arXiv:2006.03654*.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *NIPS*.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246.
- Rob Johnson, Anthony Watkinson, and Michael Mabe. 2018. The STM report. *An overview of scientific and scholarly publishing. 5th edition October*.
- Esther Landhuis. 2016. Scientific literature: information overload. *Nature*, 535(7612):457–458.
- Jiaju Lin, Jiawei Liu, Jing Ling, Zhiwei Wang, Qin Chen, and Liang He. 2021. ECNUICA at SemEval-2021 Task 11: Schema based Information Extraction Pipeline. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.
- Haoyang Liu, Janina Sarol, and Halil Kilicoglu. 2021. UIUC.BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xinge Ma, Jin Wang, and Xuejie Zhang. 2021. YNU-HPCC at SemEval-2021 Task 11: Using a BERT Model to Extract Contributions from NLP Scholarly Articles. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.

Paolo Manghi, Claudio Atzori, Alessia Bardi, Jochen Shirrwagen, Harry Dimitropoulos, Sandro La Bruzzo, Ioannis Foufloulas, Aenne Löhden, Amelie Bäcker, Andrea Mannocci, Marek Horst, Miriam Baglioni, Andreas Czerniak, Katerina Kiatoropoulou, Argiro Kokogiannaki, Michele De Bonis, Michele Artini, Enrico Ottonello, Antonis Lempesis, Lars Holm Nielsen, Alexandros Ioannidis, Chiara Bigarella, and Friedrich Summan. 2019. [OpenAIRE Research Graph Dump](#).

Anna Martin and Ted Pedersen. 2021. Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A Python natural language processing toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Shashank Shailabh, Sajal Chaurasia, and Ashutosh Modi. 2021. KnowGraph@IITK at SemEval-2021 Task 11: Building Knowledge Graph for NLP Research. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Han Wu, Yuxiao Dong, and Anshul Kanakia. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413.

Genyu Zhang, Yu Su, Changhong He, Lei Lin, Chengjie Sun, and Lili Shan. 2021. ITNLP at SemEval-2021 Task 11: Boosting BERT with Sampling and Adversarial Training for Knowledge Extraction. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). ACL.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. 2020. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, pages 11278–11287. PMLR.

## A Per Information Unit Evaluations

Table 7 shows triple extraction F1 scores for each of the IUs. The scores from each of the three evaluation phases in our Shared Task are separated by slash symbols. Recall that from the second evaluation phase, the gold data were made available to the systems starting with sentences (*Evaluation Phase 2, Part 1: Phrases and Triples*) followed by the terms and relations additionally (*Evaluation Phase 2, Part 2: Triples*).

Comparing the performances across IUs, we see the CODE IU was the easiest to extract. In *Phase 1*, the best F1 was 83.33%. In both *Phase 2, Part 1* and *Part 2*, the best F1 was 100.0%. This is an expected result for CODE to be easiest to extract since it had the simplest annotation patterns; an example is depicted in Fig. 5.

		Training Data	Test Data
1	CODE	1.0	1.03
2	RESEARCHPROBLEM	2.78	2.13
3	DATASET	13.78	22.5
4	APPROACH	15.55	17.61
5	MODEL	18.06	20.14
6	ABLATIONANALYSIS	18.82	21.19
7	BASELINES	20.13	15.3
8	RESULTS	23.31	23.0
9	HYPERPARAMETERS	23.78	19.78
10	EXPERIMENTALSETUP	27.53	27.47
11	TASKS	34.63	-
12	EXPERIMENTS	54.65	39.06

Table 6: Average no. of triples per Information Unit

Table 6 shows the average number of triples per IU reflecting, in a sense, their complexity. We hypothesize that the more the triples, the more complex the extraction task. Comparing these numbers with the results in Table 7, we see that 5th ranked IU, i.e. MODEL, showed the next easiest to extract after CODE, at 38.14% F1, in the end-to-end setting. Following which, we see that the 2nd ranked IU, i.e. RESEARCHPROBLEM, obtained an F1 of 35.79%. Nevertheless, confirming our hypothesis, we found a negative correlation ( $r -0.65$ ) between the training data triples size per IU and the end-to-end system performances, i.e. for IUs with fewer triples the extraction score is higher for most IUs. The negative correlations were progressively stronger from *Part 1* to *Part 2* in *Evaluation Phase 2* ( $r -0.75$  and  $r -0.79$ ), respectively.

Team	RESEARCHPROBLEM	APPROACH	MODEL	CODE
UIUC_BioNLP	26.17/ 53.19/ 89.41	<b>11.54/ 20.2/ 28.87</b>	<b>38.14/ 55.31/ 76.9</b>	57.14/ 80.0/ <b>100.0</b>
ITNLP	<b>35.79/ 43.18/ 78.35</b>	0.0/ 0.0/ 0.0	16.03/ 22.65/ 51.42	<b>83.33/ 100.0/ 100.0</b>
KnowGraph@IITK	24.62/ 25.81/ <b>97.56</b>	4.94/ 5.93/ 0.0	8.2/ 19.18/ 34.48	<b>83.33/ 100.0/ 100.0</b>
ECNUICA	6.45/ <b>65.12/ 89.89</b>	1.75/ 17.83/ <b>28.93</b>	0.0/ 29.73/ 56.67	0.0/ 80.0/ 80.0
INNOVATORS	9.88/ 9.88/ 3.25	0.0/ 0.0/ 3.8	0.0/ 0.0/ 7.55	50.0/ 50.0/ 0.0
DULUTH	0.0/ 4.71/ 58.73	0.0/ 2.06/ 21.78	7.23/ 7.14/ 35.36	0.0/ 40.0/ 88.89
YNU-HPCC	-/ 2.9/ 5.07	-/ 2.53/ 7.22	-/ 3.52/ 18.29	-/ 0.93/ 0.56

Team	EXPERIMENTALSETUP	HYPERPARAMETERS	BASELINES
UIUC_BioNLP	<b>28.37/ 52.42/ 67.27</b>	<b>5.6/ 35.71/ 39.44</b>	<b>20.69/ 50.85/ 74.34</b>
ITNLP	18.78/ 29.87/ 42.16	4.0/ 6.06/ 12.63	0.0/ 16.67/ 27.69
KnowGraph@IITK	12.14/ 13.53/ 12.7	4.37/ 7.88/ 8.48	3.47/ 6.78/ 33.33
ECNUICA	14.79/ 25.88/ 42.34	3.36/ 13.4/ 3.36	9.11/ 34.62/ 51.06
INNOVATORS	0.0/ 0.0/ 0.0	0.0/ 0.0/ 0.0	0.0/ 0.0/ 0.0
DULUTH	1.54/ 6.33/ 30.47	4.04/ 7.89/ 21.43	5.56/ 3.23/ 17.5
YNU-HPCC	-/ 4.24/ 16.7	-/ 0.88/ 2.58	-/ 0.87/ 3.5

Team	RESULTS	EXPERIMENTS	ABLATIONANALYSIS
UIUC_BioNLP	<b>20.62/ 37.77/ 56.4</b>	7.19/ <b>8.96/ 10.61</b>	<b>23.01/ 31.78/ 61.36</b>
ITNLP	8.85/ 17.47/ 42.5	1.48/ 1.42/ 0.0	8.6/ 6.35/ 11.63
KnowGraph@IITK	10.86/ 17.55/ 28.94	3.33/ 1.96/ 0.0	4.23/ 3.74/ 35.16
ECNUICA	15.37/ 26.25/ 49.0	<b>7.86/ 6.06/ 13.86</b>	3.94/ 4.6/ 8.82
INNOVATORS	0.0/ 0.0/ 7.36	0.0/ 0.0/ 0.0	0.0/ 0.0/ 0.0
DULUTH	7.2/ 8.04/ 30.96	0.0/ 1.72/ 5.97	0.0/ 0.0/ 12.29
YNU-HPCC	-/ 3.56/ 16.63	-/ 0.68/ 2.73	-/ 1.05/ 2.79

Table 7: Per Information Unit F1 scores per evaluation phase of the seven participating teams. The three scores in each row are from the three evaluation phases in the Shared Task as follows [Evaluation Phase 1: End-to-end Pipeline]/[Evaluation Phase 2, Part 1: Phrases and Triples]/[Evaluation Phase 2, Part 2: Triples]. Best scores are in bold.

Model	Information Units			Triples		
	F1	P	R	F1	P	R
UIUC_BioNLP	<b>72.93/83.98</b>	66.67/76.77	80.49/92.68	<b>22.28/25.01</b>	22.3/25.08	22.26/24.94
ITNLP	<b>72.93/82.49</b>	66.67/76.84	80.49/89.02	13.79/14.26	13.39/13.98	14.23/14.56
KnowGraph@IITK	60.54/72.32	44.13/57.04	96.34/98.78	8.57/10.0	6.53/7.87	12.45/13.72
ECNUICA	54.05/56.76	42.86/45.0	73.17/76.83	6.78/6.72	4.28/4.24	16.29/16.12
INNOVATORS	71.72/80.0	82.54/92.06	63.41/70.73	0.97/0.97	14.29/14.29	0.5/0.5
DULUTH	64.41/77.11	60.0/76.19	69.51/78.05	3.94/4.05	9.2/10.42	2.51/2.51

Table 8: Evaluation Phase 1: End-to-end Pipeline Results with (APPROACH, MODEL) IUs normalized to APPROACH and (EXPERIMENTALSETUP, HYPERPARAMETERS) IUs normalized to EXPERIMENTALSETUP. Best scores are in bold. Scores before the slash are from original dataset and scores after the slash are from the normalized dataset.

```
{
  "Code" : ["http://github.com/dalab/deep-ed",
    {"from sentence" : "Our code and data are
    publicly available : http://github.com/dalab
    /deep-ed"}]
}
```

Figure 5: Annotated data in JSON format for the CODE Information Unit for the paper “Deep Joint Entity Disambiguation with Local Neural Attention.”

i.e. APPROACH and MODEL as APPROACH and EXPERIMENTALSETUP and HYPERPARAMETERS as EXPERIMENTALSETUP. By this, we can observe system performances on a simplified version of our task. Observing “Triples” F1, we see that the ordering of the system performance without and with normalization remain unchanged - the best score obtained a 3 points boost.

## B Normalized APPROACH and EXPERIMENTALSETUP Evaluations

In Table 8, we revisit overall scores from Table 5 for two evaluation aspects in the end-to-end system evaluations, i.e. only for extracting Information Units and Triples. We revisit just these two aspects because they were impacted when we obtained normalizations of four IU labels into two, respectively,

# UIUC\_BioNLP at SemEval-2021 Task 11: A Cascade of Neural Models for Structuring Scholarly NLP Contributions

Haoyang Liu, Janina Sarol, and Halil Kilicoglu

School of Information Sciences, University of Illinois at Urbana-Champaign

{h157, mjsarol, halil}@illinois.edu

## Abstract

We propose a cascade of neural models that performs sentence classification, phrase recognition, and triple extraction to automatically structure the scholarly contributions of NLP publications in English. To identify the most important contribution sentences in a paper, we used a BERT-based classifier with positional features (Subtask 1). A BERT-CRF model was used to recognize and characterize relevant phrases in contribution sentences (Subtask 2). We categorized the triples into several types based on whether and how their elements were expressed in text, and addressed each type using separate BERT-based classifiers as well as rules (Subtask 3). Our system was officially ranked second in Phase 1 evaluation and first in both parts of Phase 2 evaluation. After fixing a submission error in Phase 1, our approach yielded the best results overall. In this paper, in addition to a system description, we also provide further analysis of our results, highlighting its strengths and limitations. We make our code publicly available at <https://github.com/Liu-Hy/nlp-contrib-graph>.

## 1 Introduction

With the deluge of scientific publications in recent years, keeping pace with the literature and managing information overload have become increasingly challenging for researchers. There is a growing need for tools that can automatically extract and structure semantic information from scientific publications to facilitate advanced approaches to information access and knowledge curation (Shen et al., 2018).

The field of natural language processing (NLP) has witnessed an enormous growth in recent years with advances in deep learning, and there are increasing efforts in developing methods to extract scholarly knowledge from NLP pub-

lications (QasemiZadeh and Schumann, 2016; D’Souza and Auer, 2020b). One such effort is NLPContributions, an annotation scheme for describing the scholarly contributions in NLP publications and a corpus annotated using this annotation scheme (D’Souza and Auer, 2020b). This corpus has been proposed for training and testing of machine reading models, whose output can be integrated with the Open Research Knowledge Graph framework (ORKG) (Jaradeh et al., 2019). ORKG formalizes the research contributions of a scholarly publication as a knowledge graph, which can further be linked to other publications via the graph. The goal of the NLPContributionGraph (NCG) shared task (D’Souza et al., 2021) is to facilitate the development of machine reading models that can extract ORKG-compatible scholarly contribution information from NLP publications. The shared task consists of three subtasks (see D’Souza et al. (2021) for a more detailed description):

- Subtask 1: Identification of contribution sentences from NLP publications
- Subtask 2: Recognition of scientific terms and relations in contribution sentences
- Subtask 3: Extraction and classification of triples that pair scientific terms with relations

In this paper, we describe our contribution to NCG shared task. We built a cascade of neural classification and sequence labeling models based on BERT (Devlin et al., 2019). For subtask 3, we characterized triples based on whether and how their elements are expressed in text, and employed different models for each category. We also explored rule-based heuristics to improve model performance. Our models had the best overall performance in the shared task (57.27%, 46.41%, and 22.28%  $F_1$  score in subtasks 1, 2, and 3, respectively). The results are encouraging for extracting

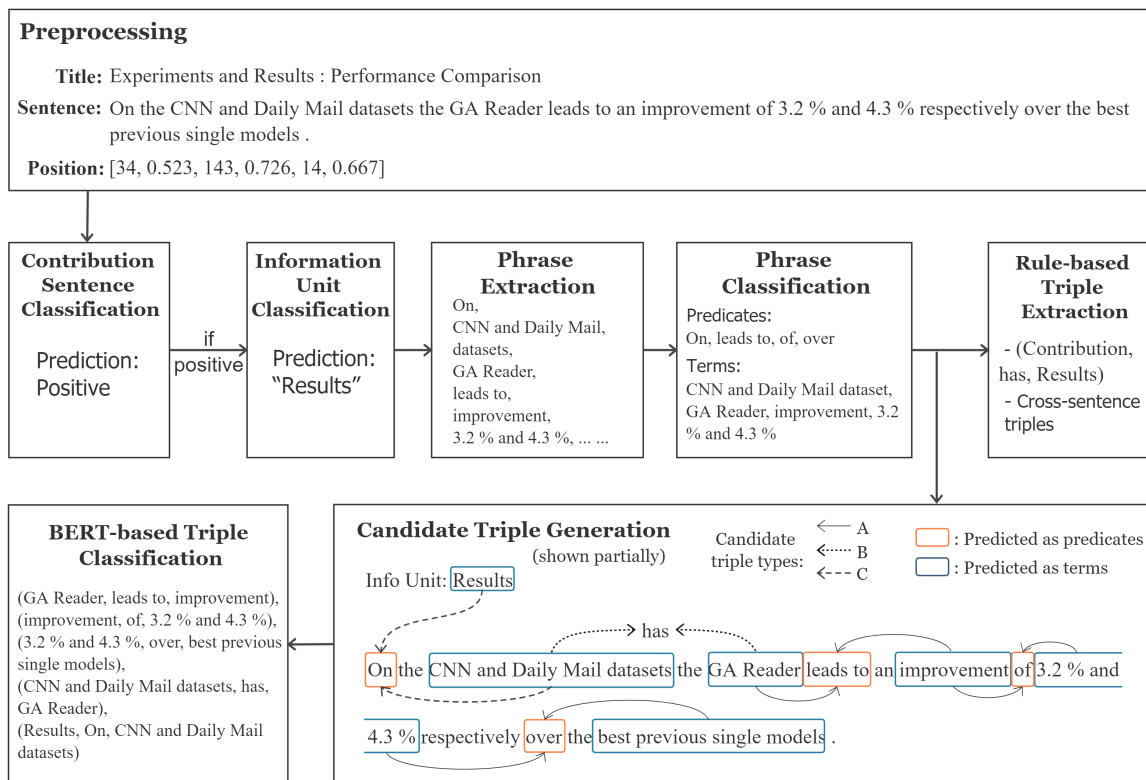


Figure 1: End-to-end system diagram.

scholarly contributions from scientific publications, although there is much room for improvement.

## 2 System Overview

In this section, we first describe our data preprocessing steps. Next, we discuss our models for each subtask, and the experimental setup for our end-to-end system (Phase 1). We provide an overview of the system in Figure 1 and provide examples for illustration, when necessary.

### 2.1 Data preprocessing

The participants of the shared task were provided three kinds of input: a) plain text files of the publications converted from PDF using Grobid<sup>1</sup>, b) sentences and tokens identified using Stanza (Qi et al., 2020), and c) triples and source texts organized by their information units (e.g., APPROACH) in JSON format.

#### 2.1.1 Identifying headers and positional information

One major preprocessing step was to identify section headers in the publications and associate them

with individual sentences. For sentence classification (subtask 1), we incorporated the topmost and innermost section headers associated with a sentence into its representation. The topmost header indicates the general role that a sentence plays in the article, while the innermost header provides more specific context for the sentence. For example, one topmost/innermost header pair is EXPERIMENT/DATA SET AND EXPERIMENT SETTINGS.

In the absence of explicit section information in the input, we used rule-based heuristics to extract these headers. With the first heuristic (Heuristic1), we simply identified the sentences following blank lines in plain text files as section headers. In Heuristic2, we first identified candidate headers as sentences that contain fewer than 10 words, have the first letter capitalized, do not end with several stopwords (*by, as, in, that, or and*), do not contain question marks in the middle or end with some punctuation (comma, colon or full stop). Next, we determined the case format used for headers in the publication by counting the occurrences of each case format type (e.g., all uppercase: EXPERIMENTAL SETUP). Among the headers that conform to the determined case format, we dis-

<sup>1</sup><https://grobid.readthedocs.io/>

tinguished topmost headers as those that contain several lexical cues (e.g., *background, method*) and are shorter than 5 words. Finally, we associated each sentence with the nearest preceding topmost and innermost header.

To incorporate headers into the sentence representation, we join the topmost and innermost header together with a colon between them and refer to it as the “title” of the sentence. In the case where a sentence is directly governed by a top-level header or it is a header itself, the title consists of the topmost header only.

We characterize the position of each sentence in the document with a combination of numeric features:

- The offset of the sentence in the entire paper.
- The offset of the sentence with respect to its topmost header.
- The offset of the sentence with respect to the header extracted using Heuristic 1.

Each of these offset features are divided by the number of sentences in the corresponding discourse (entire paper or the section) to extract a proportional sentence position feature. Thus, for every sentence, a total of six positional features (three offsets, three proportional sentence positions) are computed.

### 2.1.2 JSON Parsing

We created two additional models to assist with triple extraction: a) a multi-class sentence classifier that labels each sentence with a single information unit and b) a binary phrase classifier that labels phrases as scientific terms vs. predicates (described below). To train these models, we extracted additional information from JSON files. First, we matched the contribution sentences with the source text in the JSON files to get the information unit labels of the sentences. Second, we aligned the phrases with the triples in the same information unit, and determined whether each phrase is a predicate or term based on its location in the triple.

## 2.2 Subtask 1: Contribution Sentence Classification

We built a binary classifier to determine whether each sentence describes a contribution of the publication. Our analysis revealed that this decision was not simply based on the semantics of the sentence, but also its position in the document. On

one hand, the section header associated with the sentence provides important clues about the role of the sentence in the larger context. For example, the header “Related Work” indicates that sentences in this section are likely to discuss the contributions of prior research. On the other hand, some parts of the documents are naturally more salient than others (e.g. title, abstract, the first few lines of each section), where authors tend to summarize the most important information. To operationalize these insights, we designed a model that captures the information about the sentence, its topmost and innermost headers as well as its position in the document, as discussed above.

We used a BERT model to encode the sentence and its title (i.e., concatenated headers) separately and concatenated their textual representation together with the positional features to obtain a sentence representation. We then fed this representation into two dense layers, and used a final softmax layer for classification (Figure 2).

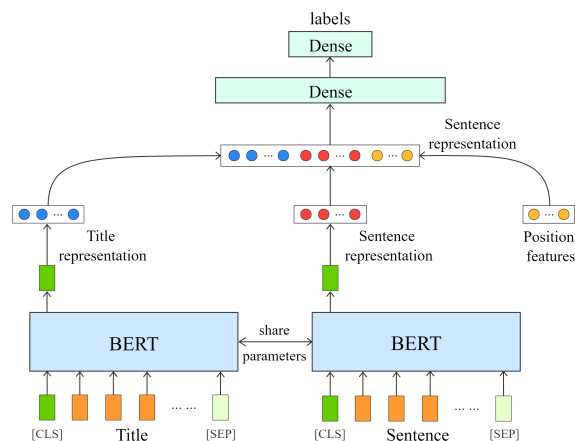


Figure 2: Sentence classification model architecture

## 2.3 Subtask 2: Phrase Recognition

Subtask 2 is similar to a named entity recognition (NER) task, although the participating systems were only required to extract relevant text spans and not to categorize them. One major difficulty with this subtask is that phrases do not align neatly with sentence constituents (e.g., noun phrases) and they vary greatly in length and in what counts as their boundaries (e.g. *best results* and *our best results* are both valid phrases).

For this subtask, we used a BERT-CRF model for phrase extraction and type classification (Souza et al., 2019). The raw text of the sentence is taken

as the model input. A BIO scheme that incorporates phrase types (scientific term vs. predicate) is used (e.g., B-Predicate, I-Term, O). The probabilities produced by the BERT model are fed into a Conditional Random Field (CRF) layer (Lafferty et al., 2001) for end-to-end training. We note that while phrase type classification is not necessary for subtask 2, we perform it since it is useful for our subtask 3 model, described next.

## 2.4 Subtask 3: Triple Extraction

Subtask 3 involves organizing phrases into triples. In information extraction, semantic triples are typically composed of subject, predicate, and object terms each corresponding to specific textual spans. This is not always the case in this subtask. While in most cases all three terms are extracted from a single sentence, a non-negligible number of triples consist of at least one phrase that does not come from the sentence (e.g. (TASKS, has, *Coreference resolution*), where the subject is an information unit and the predicate is not a sentence element).

To better understand triple characteristics, we categorized them into several types based on their composition, and created separate relation classification models for each type. The triple categorization is presented in Table 1. For each type, we list their functions in information organization, their proportion to all triples, along with some examples. We note that input to the training process for triple extraction varies by the type of the triple (described for each type in Section 2.4.2).

### 2.4.1 Information Unit Classification

To aid triple extraction, we modified the binary classification model that we trained for subtask 1 to further classify contribution sentences by their information units (multi-class classification). The process of labeling contribution sentences with information units was briefly described in Subsection 2.1.2.

In analyzing the information units, we identified two special pairs (MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS). In the dataset, no document contains both units of a pair. The decision of which unit to choose is made at the document level. Therefore, we merged the labels of similar units before feeding the examples into the multi-class classification model.

After classification, we used lexical rules to split these units. Our rules were based on the following

observations. First, the MODEL vs. APPROACH distinction seems related to how the authors mention their work in the abstract and section headers of the paper. Second, EXPERIMENTAL-SETUP is often used instead of HYPERPARAMETERS when the hardware or the framework used in the study is specified (e.g. *V100 GPU, Pytorch*).

We did not recognize CODE information units using this model, since we found that such sentences can be identified with a very high accuracy using a simple rule based on presence of a URL in the sentence.

### 2.4.2 Neural models for triple extraction

We extract triples of type A, B, C and D (Table 1) by formulating them as neural relation classification tasks. All the classifiers are vanilla BERT classifiers (one linear layer followed by softmax). For each type, we observed the patterns in the training data, and addressed the most common ones. Ignoring the less frequent patterns inevitably led to a lower recall ceiling in our models.

**Type A** This type, in which all triple elements are mentions in the sentence, represents the majority of the triples. The corresponding model classifies the triples as a whole (“triple classification”). To the best of our knowledge, little research has been done on relation classification among three phrases; however, the Transformer model at the core of BERT is versatile enough to succeed in a wide range of tasks. As our training examples, we take every combination of a predicate and two terms in a sentence as a candidate triple, and train a model that predicts whether the three phrases constitute a triple or not. We encode the relation between three phrases by marking their boundaries in the sentence, as shown in Example 1. We use angle brackets to enclose predicates, and square brackets to enclose terms.

- (1) *In this paper* , we explore an alternate `[[ semisupervised approach ]]` which does `<< not require >> [[ additional labeled data ]]`.

**Type B** To identify triples of type B (two terms from the sentence and the relation type one of has, name, or None), we classify the relation between each pair of terms in a sentence that are not related by a type A triple. We found that 96% of these triples preserve the order of the two terms in the sentence, so we also preserve the order for extraction.



	Composition	Examples	Role	Pct.
Type A	Three phrases in a sentence	( <i>Deep - ED</i> , obtain, <i>BLEU score</i> )	Organize the semantics of a sentence.	57%
Type B	Two terms in a sentence with an added predicate <i>has</i> or name	( <i>ByteNet Decoder</i> , has, <i>30 residual blocks</i> )	Organize the semantics of a sentence.	7%
Type C	Information unit (subject), and two phrases in a sentence (predicate and object)	(HYPERPARAMETERS, use, <i>cross - entropy loss</i> )	Link a sentence to its information unit.	9%
Type D	Information unit (subject), <i>has</i> (predicate), and a term in the sentence (object)	(HYPERPARAMETERS, has, <i>starting learning rate</i> )	Link a sentence to its information unit.	9%
Type E	CONTRIBUTION (subject), <i>has</i> (predicate), information unit (object) OR CONTRIBUTION (subject), fixed (predicate), and a phrase (object) for the information units RESEARCH PROBLEM and CODE	(CONTRIBUTION, has, RESULTS), (CONTRIBUTION, has research problem, <i>neural machine translation</i> )	Link the “Contribution” node of each paper to an information unit.	9%
Type F	Cross-sentence triples	( <i>Positional Encoding</i> , inject, <i>some information</i> )	Structure the information across sentences	3%

Table 1: Triple types, their roles, and frequency. Types A-D are addressed using neural models and Types E-F with rules. 6% of triples do not fit in these categories and are not shown.

**Type C** Type C triples involve an information unit name as the subject along with a predicate and object from the sentence. We found that 89% of these triples take the first predicate and the first term in a sentence as their predicate and object respectively. Furthermore, in 98% of these sentences, the first predicate precedes the first term. Therefore, we classify each sentence whose first predicate precedes the first term, to predict whether a triple of this type can be extracted from the sentence. To train this classifier, we prepend the information unit name to the sentence text with a colon in between, as in Example 2 (*Model* is the information unit).

(2) *[[ Model ]]: In this work , we << introduce >> [[ a new type of linear connections ]]* for multi - layer recurrent networks .

**Type D** Type D triples are similar to Type C, but instead of a predicate phrase from the sentence, they involve the non-sentence predicate *has*. We found that 95% of these triples in the training set take the first term in the sentence as their object, and the first predicate in the sentence, if one exists, almost always follows the first term. Therefore, we classify each sentence that conforms to this pattern, to predict whether the information unit name and the first term constitute a *has* relation. We prepend the info unit name to the sentence in the same way

as in Type C.

### 2.4.3 Rule-based triple extraction

Triples of type E and F are extracted using heuristic rules. For type E, the subject is always CONTRIBUTION. The predicate can be *has*, in which case the object is the name of an information unit. If the related information unit is CODE or RESEARCH PROBLEM, the predicate is a fixed predicate (*Code* or *has research problem*, respectively) and the object is a phrase from the sentence. These rules use phrase and information units identified in earlier steps (Sections 2.3 and 2.4.1, respectively).

We developed the following rules to extract cross-sentence triples (type F):

1. If the first sentence has a single entity, and the second sentence has at least 2 entities, we assign the entity in sentence 1 as the subject and the first and second entities in sentence 2 as the predicate and object, respectively. We add this triple to the list only if both subject and predicate are noun phrases, which prevents many false positives. We also add the corresponding triple in the form of INFO-UNIT-*has-subject* (e.g. MODEL-*has-Encoder*). In many sentences that follow this rule, the first sentence is a section header.

	Avg F <sub>1</sub>	Information Units			Sentences			Phrases			Triples		
		F <sub>1</sub>	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R
Our system	49.72	72.93	66.67	80.49	57.27	53.61	61.46	<b>46.41</b>	42.69	50.83	<b>22.28</b>	22.30	22.26
IAA	52.82	<b>79.73</b>	78.83	80.65	<b>67.44</b>	67.25	67.63	41.84	45.36	38.83	<b>22.28</b>	23.76	20.97

Table 2: End-to-end performance (Evaluation Phase 1). IAA: intra-annotator agreement.

- If the two sentences each contain a single term and sentence 1 term is a substring of sentence 2 term or if sentence 1 term is an acronym of sentence 2 term, we create the following triple: *term 1-name-term 2*. We extract a term’s acronym by combining the initials of each token in the entity. An example of a term pair that follows this rule is (*GLUE, General Language Understanding Evaluation*).

These rules are applied to consecutive sentences only. In the training set, we found 812 triples that follow these rules, 649 (80%) of which could be identified correctly using these rules.

## 2.5 Experimental Setup

We implemented our models using Simple Transformers<sup>2</sup>. We used SciBERT (Beltagy et al., 2019) as the pre-trained language model. To train our models, we used a batch size of 16, and empirically found the best learning rate for each model between  $10^{-5}$  and  $10^{-4}$ . One exception was that in our sentence classification model (subtask 1), we used a fixed learning rate of  $10^{-5}$  to fine-tune the BERT, and a larger learning rate between  $5 \times 10^{-5}$  and  $10^{-3}$  for the dense layers. We used the AdamW optimizer (Loshchilov and Hutter, 2017) and the polynomial decay scheduler with the power of 0.5. We ran the experiments on a Google Cloud VM instance, using a Tesla V100 GPU.

## 3 Results

All the subtasks were evaluated on F<sub>1</sub> scores, and among them, triple extraction is evaluated by the micro-average of F<sub>1</sub> scores on each information unit. In the end-to-end evaluation (Phase 1), the participants were provided with the raw input to perform all three subtasks sequentially. We were officially ranked second in Phase 1, due to a submission error that resulted in phrase extraction F<sub>1</sub> of zero. Our correct submission achieved an average F<sub>1</sub> of 49.7%, the best score among all participating teams. Table 2 shows our performance in Phase 1,

<sup>2</sup><https://github.com/ThilinaRajapakse/simpletransformers>

and the intra-annotator agreement (IAA) on each subtask (D’Souza and Auer, 2020a).

We observe that, although the performance of our system on sentence classification is lower than human performance (57.27% vs. 67.44% F<sub>1</sub>), using its own sentence predictions, our system outperforms human annotators on phrase recognition (46.41% vs. 41.84% F<sub>1</sub>), and reaches comparable performance to human annotators on triple extraction. We also note that our system generally performs better in terms of recall than precision.

We were officially ranked first in both parts of Evaluation Phase 2. In Part 1, the participants were provided with the sentences labels to conduct phrase recognition and triple extraction sequentially; in Part 2, both the sentence labels and the phrase labels were provided to extract triples. We essentially followed our method in Phase 1 on phrase recognition and triple extraction, but made several attempts to improve the performance, which we discuss in Section 4. Our results in both parts of the Phase 2 evaluation are shown in Table 3. Compared to Phase 1 evaluation, we observe a significant improvement in phrase recognition (46.41% vs. 78.57% F<sub>1</sub>) in Part 1 and in triple extraction (22.28% to 43.44% and 61.29% F<sub>1</sub>) when ground truth contribution sentences and phrases are provided.

## 4 Performance Analysis

In this section, we analyze the performance of several components of our system and compare different schemes for entity representation and triple extraction. We also discuss some possible methods for improvement based on our shared task results and follow-up experiments.

### 4.1 Contribution Sentence Classification

We conducted ablation experiments to evaluate the effect of features for contribution sentence classification. Table 5 shows the model performance on the 10% validation set when using all features, using either the title or the position features together with the sentence, and using the sentence only.

	Information Units			Phrases			Triples		
	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R
Part 1	82.49	76.84	89.02	78.57	76.86	80.35	43.44	45.06	41.94
Part 2	82.49	76.84	89.02	-	-	-	61.29	65.19	57.82

Table 3: Performance in phrase and triple extraction (Evaluation Phase 2). Note that we focused only on triple extraction in Part 2, therefore the information unit extraction performance remains the same.

Unit name	Research problem	Approach	Model	Code	Dataset	Experimental Setup	Hyperparameters	Baselines	Results	Tasks	Experiments	Ablation analysis
F <sub>1</sub>	94.64	24.14	86.22	87.50	80.00	58.29	72.61	91.45	94.65	90.48	83.16	90.68

Table 4: Information unit classification performance.

Settings	F <sub>1</sub>	P	R
Sentence + title + position	65.11	63.96	66.30
Sentence + title	63.87	61.00	67.03
Sentence + position	52.28	46.38	59.89
Sentence only	51.39	49.00	54.03

Table 5: Results of ablation experiments on contribution sentence classification task.

We observe the title information significantly improves the performance, and the position features are also helpful, to a lesser extent. Combining the title and the position features gives the best performance on contribution sentence classification.

## 4.2 Information Unit Classification

In Evaluation Phase 2, the ground truth labels for contribution sentences increased the performance of our base model on information unit classification from 72.93% to 76.84% F<sub>1</sub>. To further improve our method, we ensembled 45 multi-class sentence classifiers by averaging their output (using *bagging*), which increased the F<sub>1</sub> score to 78.65%. Next, we improved our rules for distinguishing the special pairs (MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS) by adjusting the lexical cues with more careful observation of the data, which results in our final performance (82.49% F<sub>1</sub> in Table 3).

For further analysis, we evaluated the classification performance on each information unit, as shown in Table 4. The related confusion matrix is shown in Fig. 3. We observe that severe confusion mainly occurs between MODEL vs. APPROACH and EXPERIMENTAL-SETUP vs. HYPERPARAMETERS, pairs that we grouped together in neural classification. This shows that while our sentence classification model has good accuracy, there is still much room for improvement in the rule-based differentiation of similar units.

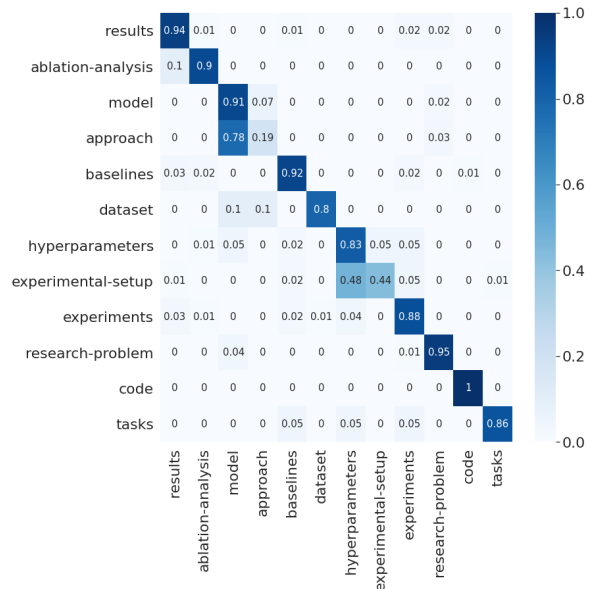


Figure 3: Confusion Matrix

The differentiation between MODEL and APPROACH is particularly challenging. While some papers aim at discussing an abstract idea and some focus on system implementation, most papers fall in the gray area between them. We also attempted neural classification on the abstracts to deal with this issue, but the result were not satisfactory.

## 4.3 Phrase extraction and classification

**Specific BIO VS. simple BIO** Alternative to our method of using specific BIO tags to indicate phrase types (Subsection 2.3), we also used another scheme (“simple BIO”), in which we only used (B, I, O) tags to mark phrase boundaries.

With this scheme, we first trained a BERT-CRF model to extract the phrases, and then trained a binary BERT classifier to predict phrase types. The sentence along with the phrase marked by special tokens is fed into the BERT model for binary clas-

Settings	Phrase extraction			Phrase classification		
	F <sub>1</sub>	P	R	F <sub>1</sub>	P	R
Specific BIO	76.09	75.57	76.62	98.13	98.00	98.25
Simple BIO	77.13	76.33	77.95	98.27	98.70	97.85
Simple BIO + ensemble	78.57	76.86	80.35			

Table 6: Phrase extraction and classification performance. We take *predicate* as the positive label to calculate the F<sub>1</sub> score for phrase classification.

sification. The performance comparison of these schemes is shown in Table 6. While both schemes are effective, simple BIO outperforms specific BIO in phrase extraction by a small margin, so we used this scheme in Evaluation Phase 2.

The difference may be due to the noise in phrase types. Specifically, there is a good number of gerund phrases, on which the predicate-term differentiation is challenging. Moreover, in some cases, a verb phrase is used as a term to form triples. Combined with the relatively low intra-annotator agreement, these observations suggest that uncertainty and noise in the data affects the performance of the models. Note that the specific BIO scheme eliminates the need for a separate phrase classification model, making it preferable when the training and inference speed is a concern.

**Error analysis and improvement** We investigated the wrong predictions of our phrase extraction model, and found that most errors are due to boundary detection issues. For example, in one sentence, the model predicts *all layers of representation* as a phrase, while *all layers, of, representations* are annotated as three separate phrases. The opposite situation also occurs, when the model predicts a single unit as separate phrases. Another type of boundary error occurs when the model cannot predict correctly whether to include a non-core phrase element, like an adverb, in the phrase or not (e.g., it predicts *see that* whereas the annotated phrase is *also see that*). We believe that a relaxed boundary match evaluation can be considered for this task.

We attribute these errors to the uncertainty in semantic granularity, and attempted to alleviate the problem by ensembling. We get 12 bootstrap samples from the training data, and on each sample, we train the model and save its snapshot after each epoch from the 3th epoch to the 10th epoch, to get a total of 96 submodels. To aggregate their predictions, we extract a phrase in a sentence only if it is predicted by more than N submodels, where

N is a hyperparameter around 48. We present the result in Table 6 for comparison. We observe that ensembling noticeably improved phrase extraction (from 77.13% to 78.57% F<sub>1</sub>).

#### 4.4 Triple extraction

**Triple vs. pairwise classification** In addition to triple classification method (Subsection 2.4.2) to extract type A triples, we also used pairwise classification for this task. In this scheme, we considered every (subject, predicate, object) triple as a composition of two (predicate, term) pairs, or “candidate pairs”, and used a neural model to predict whether the two phrases in the pair are associated. After prediction, we reconstructed triples from the predicted pairs using rules. If a predicate is predicted to be associated with two terms, we combine them into a triple while preserving the order of the two terms in the sentence (subject first). If one predicate is associated with more than two terms, we only extract the triples in which the predicate is located between the two terms in the sentence. With only a few exceptions, we confirmed the effectiveness of these reconstruction rules; in other words, the performance of the pairwise scheme depends mainly on the classification accuracy on candidate pairs.

We compared the performance of the two schemes for type A triple extraction on the 10% validation set. We also attempted to address the imbalance of class labels resulting from both schemes by downsampling and class weight adjustment.

Settings		F <sub>1</sub>	P	R
No adjustment	Pair	91.33	91.23	91.43
	Triple	75.95	70.58	82.20
Downsampling	Pair	91.31	89.09	93.63
	Triple	80.04	79.43	80.66
Class weight	Pair	91.30	88.93	93.79
	Triple	<b>80.37</b>	81.35	79.42

Table 7: Performance of the pairwise classification scheme.

In the pairwise classification scheme (Table 7),

	F1	P	R
No adjustment	<b>87.54</b>	85.93	89.22
Downsampling	75.59	62.32	96.04
Class weight	83.35	74.94	93.89

Table 8: Performance of the triple classification scheme.

there is a 11% drop in the F1 score from the candidate pair classification to triple prediction, which is not unexpected as the model needs to correctly classify *both* of the candidate pairs to correctly predict a triple.

Table 8 shows the performance of the triple classification scheme, which achieves better performance compared to the pairwise classification scheme (87.54% vs. 80.37% F<sub>1</sub>). We also observed that the best performance was obtained without dealing with the imbalanced data. It seems that despite constituting a small portion of the dataset (9.7%), the number of the positive samples is large enough for the model to learn useful patterns.

**Type-specific performance** We also evaluated our deep learning methods for the extraction of the four types of triples, as shown in Table 9.

Type	F1	P	R
A	87.54	85.93	89.22
B	55.56	88.24	40.54
C	83.33	77.96	89.51
D	75.86	78.11	73.74

Table 9: Performance of triple extraction on each type.

Whereas our models for Type A, C, and D perform generally well, our model for Type B is far less accurate. Type B is a little special among the four types in that it requires the prediction of relation types. The type *has* is more difficult to predict than *name*, because the sentence often lacks semantic clues about the belonging or inclusion relationship between the two terms. A plausible idea is to incorporate *has* into the input, but it is difficult to do so without breaking the grammatical integrity of the sentence. We leave this improvement for future work.

**Coordination in triple extraction** A common problem we observed in our triple extraction models is the failure to account for coordination between terms. Example 3 shows a sentence with the terms in bold, and the two type C triples associating

them. Our model only extracts the first triple, and misses the second.

- (3) *The MoE consists of a **number of experts**, each a simple feed - forward neural network, and a **trainable gating network** which selects a sparse combination of the experts to process each input.*

(APPROACH, consists of, *number of experts*)

(APPROACH, consists of, *trainable gating network*)

We attempted to address this issue in post-processing, and used Stanza dependency parser (Qi et al., 2020) to detect coordination of words in phrases. If one phrase is used in a triple, we generated a parallel triple by replacing the term with the other. While this method improves recall (from 57.57% to 58.41%), it also led to precision errors (from 65.15% to 61.77%), its overall effect being negative (from 61.13% to 60.04% F<sub>1</sub>). We plan to refine this approach in future work.

## 5 Conclusion

We developed a system to generate structured representations of research contributions described in NLP publications in a manner compatible with the ORKG framework, achieving the top performance in the NCG shared task. We combined a cascade of state-of-the-art BERT-based classification and sequence labeling models with rule-based methods. In particular, we proposed a novel approach for triple extraction, where we tackled triples with different characteristics using different relation classification methods. We also explored various alternatives to the components in our end-to-end system to analyze the contribution of individual components.

In future work, we plan to improve the differentiation of similar units (e.g., MODEL vs. APPROACH), improve the extraction of type B triples, and address coordinated triples more thoroughly. We did not attempt to extract approximately 6% of the triples that did not fit in our classification (Table 1). These often involve nested information units, and we also hope to explore them in more depth in future work.

## References

- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text.

- In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: NLPContributionGraph - structuring scholarly NLP contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Jennifer D’Souza and Sören Auer. 2020a. [Graphing Contributions in Natural Language Processing Research: Intra-Annotator Agreement on a Trial Dataset](#).
- Jennifer D’Souza and Sören Auer. 2020b. [NLPContributions: An Annotation Scheme for Machine Reading of Scholarly Contributions in Natural Language Processing Literature](#).
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Ed-dine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. [Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge](#), page 243–246. Association for Computing Machinery, New York, NY, USA.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- Ilya Loshchilov and Frank Hutter. 2017. [Decoupled weight decay regularization](#). *arXiv preprint arXiv:1711.05101*.
- Behrang QasemiZadeh and Anne-Kathrin Schumann. 2016. [The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 1862–1868, Portorož, Slovenia. European Language Resources Association (ELRA).
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A python natural language processing toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.
- Zhihong Shen, Hao Ma, and Kuansan Wang. 2018. [A web-scale system for scientific knowledge exploration](#). In *Proceedings of ACL 2018, System Demonstrations*, pages 87–92, Melbourne, Australia. Association for Computational Linguistics.
- Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2019. Portuguese named entity recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*.

# KGP at SemEval-2021 Task 8: Leveraging Multi-Staged Language Models for Extracting Measurements, their Attributes and Relations

Neel Karia\*, Ayush Kaushal\*, Faraaz Mallick\*  
Department of Computer Science and Engineering  
Indian Institute of Technology Kharagpur  
{karianeel, ayushkaushal, faraazrm}@iitkgp.ac.in

## Abstract

SemEval-2021 Task 8: MeasEval aims at improving the machine understanding of measurements in scientific texts through a set of entity and semantic relation extraction sub-tasks on identifying quantity spans along with various attributes and relationships. This paper describes our system, consisting of a three-stage pipeline, that leverages pre-trained language models to extract the quantity spans in the text, followed by intelligent templates to identify units and modifiers. Finally, it identifies the quantity attributes and their relations using language models boosted with a feature re-using hierarchical architecture and multi-task learning. Our submission significantly outperforms the baseline, with the best model from the post-evaluation phase delivering more than 100% increase on F1 (Overall) from the baseline.

## 1 Introduction

Most scientific experiments are accompanied by relevant measurements, which help researchers to quantify their observations and qualitative arguments. Measurements also play a pivotal role in summarizing large experiments, and provide a brief idea of the results obtained. It is customary for scientists to present their research in the form of scientific papers. Nowadays, with thousands of papers being published digitally every year, it is extremely difficult to go through every single paper in order to get the desired data. The most popular electronic open-access repository of e-prints, arXiv, currently has 1,867,929 articles<sup>1</sup>. The sheer vastness of this number suggests just how important it is for us to automate the task of extracting measurement-related information from research papers (Singh et al., 2016).

\*Equal contribution.

<sup>1</sup>as of 9<sup>th</sup> April, 2021

A thorough understanding of the measurements not only requires the numerals, but also the context in which the quantities occur. Moreover, the entities and the properties measured along with the qualifiers that condition the measurements are crucial for understanding the measurement. MeasEval (Harper et al., 2021) is a semantic relation extraction task focused on obtaining 9 different entities pertaining to counts, measurements and qualifying attributes of these quantities in a collection of excerpts from research papers in English. Figure 1 shows an example of a quantity along with its attributes and relations from this dataset.

We propose a three-stage pipeline to address this task. The first stage uses a pre-trained BERT model (Devlin et al., 2019) to detect quantity spans from sentences. Receiving the detected spans as inputs, the second stage obtains the units and modifiers using extracted units and modifier keywords. Finally, the third stage receives the quantity spans from the first stage and uses another pre-trained language model over each quantity-span-conditioned sentence to obtain quantity-span-aware contextualized representations for each sub-token in the sentence. These representations are then used to detect the measured entity corresponding to each quantity (if any). The predictions from the measured entity task are then fused with the individual representations for each sub-token. These representations are used to detect the measured property and the qualifiers in a multi-task learning setting (Ruder, 2017).

Our submission surpassed the baseline by a significant margin and ranked 3<sup>rd</sup> for the Unit task. Our current best model delivers 516.7%, 436.8%, and 296.4% F1 (Overlap) (Mei and Radev, 1979) gains for Measured Entity, Measured Property and Qualifier tasks respectively, over the baseline.

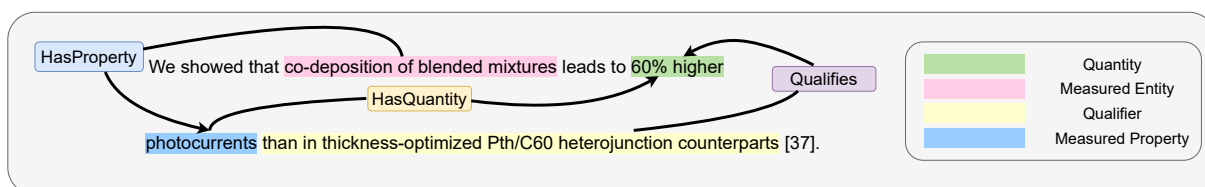


Figure 1: Visualization of Annotated Dataset

## 2 Related Works

Understanding and extracting information from scientific documents has been receiving increasing interest (Tsai et al., 2006; Nadeau and Sekine, 2007). Extracting units of measurement from scientific documents was previously studied via regular expressions and supervised classifiers (Berrahou et al., 2013; Sevenster et al., 2015).

In the orthogonal direction, there has been rapid progress in understanding natural language using deep pre-trained language models (Peters et al., 2018; Devlin et al., 2019; Liu et al., 2020; Yang et al., 2019), which has led to a general improvement across multiple tasks. The sequence labelling (Lample et al., 2016; Panchendrarajan and Amaesan, 2018) and span prediction (Luo et al., 2020; Pang et al., 2019) tasks for natural language have also received great interest recently. We build upon these systems.

## 3 Problem Statement

We are given a set of **documents**  $D = \{(d_i)_{i=1}^n\}$ . Every document  $d_i \in D$ , consists of various **Quantity** (Q) spans  $Q_i = \{(q_i^j)_{j=1}^m\}$ . Every  $q_i^j \in Q_i$ , can have a **Unit** of measurement (e.g. *cm*, *ml*) associated with it. Also every  $q_i^j \in Q_i$  is associated to some (or no) **Modifiers** (Mod) which provide information about the type of Q (e.g. whether it is a range of values, whether it denotes the Median of a set of values, etc.)<sup>2</sup>. For every  $q_i^j \in Q_i$ , there can exist a corresponding **Measured Entity** (ME)  $e_i^j$ . Some Qs do not have any ME, e.g. in ‘3413 women’, the measurement is 3413 and ‘women’ is the ‘unit’ of 3413 and **not** its ME (according to “S0006322312001096-1177.tsv”). Similarly in ‘three occasions’, the measurement is ‘three’ and ‘occasions’ is its ‘unit’ and not its ME (according to “S0165587612003680-1078.tsv”). If a  $q_i^j$  has a corresponding ME  $e_i^j$ , it can also have an associated **Measured Property** (MP)  $p_i^j$ . Finally, the

<sup>2</sup><https://github.com/harperco/MeasEval/tree/main/annotationGuidelines>

Qs, MEs and MPs can have a number of **Qualifiers** (Qual)  $qual_i^j$  providing additional information about them.

The problem also introduces three relations, namely **Qualifies** (QS), **HasProperty** (HP), and **HasQuantity** (HQ). These relations are defined between Qs, MEs, MPs and Quals as binary classification functions ( $f(x, y) \rightarrow (0, 1)$ ):

- $QS(x, a) = 1, \iff$  the Qual,  $a$ , *qualifies* the element  $x$ , where  $x$  is a Q, ME or MP.
- $HP(p, e) = 1, \iff$  the MP,  $p$ , is *associated* with the ME,  $e$ .
- $HQ(y, q) = 1, \iff$  the Q,  $q$ , is *related* to element  $y$ , where  $y$  is an ME or MP.

The problem statement consists of 5 sub-tasks. We deal with identifying all Q spans in the documents in sub-task 1, followed by detecting the Units and Mods for each identified Q in sub-task 2. In sub-tasks 3 and 4, we identify the ME, MP, and Qual spans, corresponding to the extracted Qs. Finally in sub-task 5, we identify the relationships HQ, HP, and QS between the detected Q, ME, MP, and Qual spans. Figure 1 shows the annotation procedure to be followed (Stenetorp et al., 2012).

## 4 System Overview

We model all the previously described sub-tasks as supervised learning problems. Firstly, we perform a minimal pre-processing of sentence segmentation and number normalization on the documents. Then, Stage 1 handles sub-task 1 and the Stage 2 handles sub-tasks 2 respectively, and the remaining ones are handled by Stage 3 of our pipeline.

Before proceeding to describe our approach, we describe the baseline model, provided by the task organizers. The baseline treats the detection of Q, ME, MP and Qual spans all as sequence labeling problems. It uses the spaCy Entity Tagger model (Honnibal et al., 2020) to extract all these four spans. The Units for these Qs are obtained by



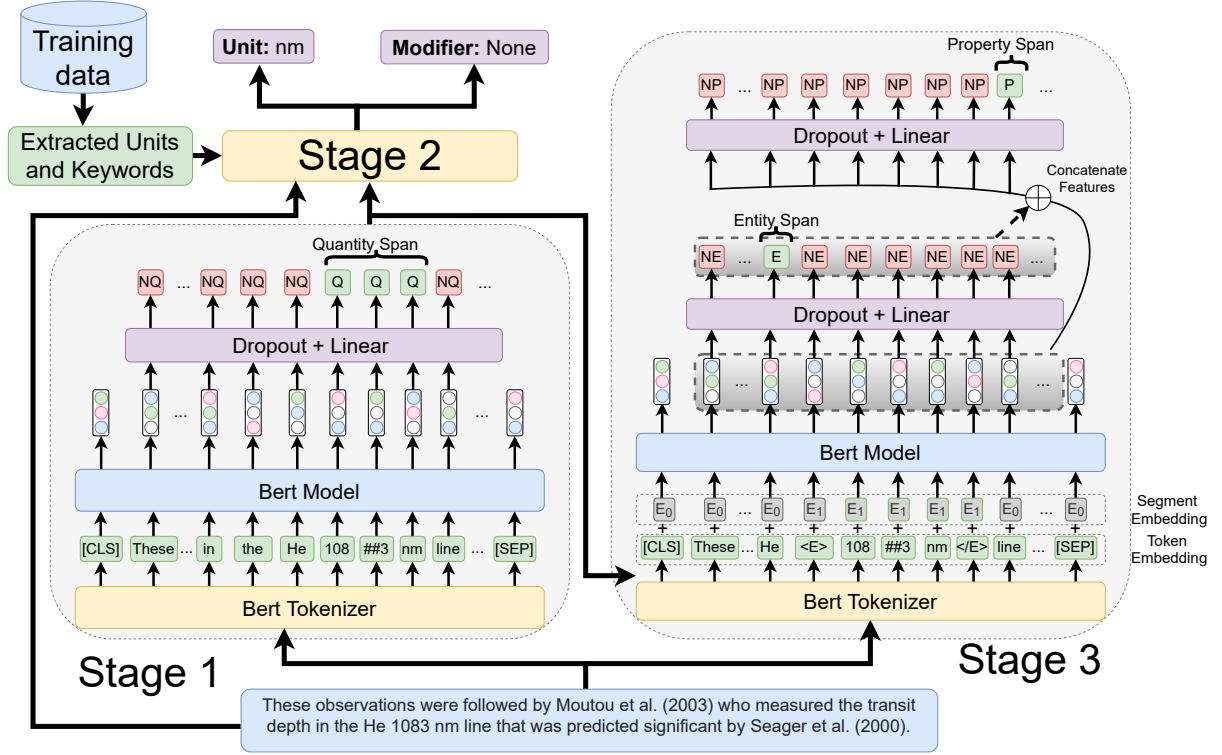


Figure 2: Overview of our Pipeline

matching the largest Units in these predicted spans with those from the train dataset.

#### 4.1 Stage 1

Similar to the baseline, we treat the Q spans learning problem as a sequence labelling problem. This is an intuitive step as it can detect multiple spans within the same text segment while being significantly cheaper in terms of the computation cost. Specifically, for a given sentence  $s$ , the input to our model is  $[CLS] s [SEP]$ . It is sub-word tokenized (Wu et al., 2016) to get the one-hot sub-token sequence  $w_0, w_1 \dots w_n$ . These sub-tokens are then fed to BERT to obtain the contextualized representations  $x_0, x_1 \dots x_n$  as follows.

First the word vectors are obtained using the Embedding  $E$  and Positional-Embedding  $E_{pos}$ :

$$x_j^{(0)} = w_j E + E_j^{pos} \quad (1)$$

Then these vectors are passed through  $L$  layers of transformer encoder (Vaswani et al., 2017) to obtain the contextualized representation. Each transformer encoder layer  $l$  receives the output vector sequence  $\{(x_j^{(l-1)})_{j=0}^n\} = x_0^{(l-1)}, x_1^{(l-1)} \dots x_n^{(l-1)}$  from the previous layer  $l - 1$  and computes the

output representation  $\{(x_j^{(l)})_{j=0}^n\}$  as follows:

$$\{(z_j^{(l)})_{j=0}^n\} = LN(MSA(\{(x_j^{(l-1)})_{j=0}^n\}) + \{(x_j^{(l-1)})_{j=0}^n\}) \quad (2)$$

$$\{(x_j^{(l)})_{j=0}^n\} = LN(\{(z_j^{(l)})_{j=0}^n\} + \{((W_2^l)^T f((W_1^l)^T z_j^{(l)} + b_1^l) + b_2^l)_{j=0}^n\}) \quad (3)$$

Here  $MSA$  is Multi-headed Self Attention and  $LN$  denotes Layer Norm.  $W_2^l, W_1^l, b_1^l, b_2^l$  are trainable parameters and  $f$  is the activation function.

The final contextualized representations  $\{(x_j)_{j=0}^n\} = \{(x_j^{(L)})_{j=0}^n\}$  are the outputs of the  $L^{th}$  transformer layer. Finally, these representations  $x_j$  (excluding  $j = 0, j = n$  for  $[CLS], [SEP]$  tokens) are each classified to a binary label:

$$(y_j^{NQ}, y_j^Q) = W_c^T x_j + b_c \quad (4)$$

Here  $W_c$  and  $b_c$  are learnable parameters and  $(y_j^{NQ}, y_j^Q)$  are the logits. This formulation of our problem can also be treated as the popular BIO tagging scheme excluding the 'B' beginning tag. This is then used to greedily match the largest contiguous span of sub-tokens with positive labels.

## 4.2 Stage 2

This stage receives the Q span predictions from Stage 1 as input and uses a method similar to the baseline, to obtain the Units. We extracted the set of Units occurring in the annotated Qs, from the documents in the dataset. However, in scientific documents, often combinations of units are present (e.g.  $Kgms^{-2}$  is a combination of ‘Kg’, ‘m’ and ‘s’). Our future work includes extending our approach to be exhaustive to handle such complex combinations of units.

To obtain the keywords for modifiers, given a Q span, we extracted the set of tokens occurring inside the span as well as in the neighboring window of 10 characters, on either side of the actual span. We discarded stopwords, punctuation marks and numbers. Then, we calculated the rate of co-occurrence between the remaining set of tokens and the Mods in the train dataset. This helped us to obtain keywords acting as significant cues for the respective Mod classes. Examples include “approximately” for IsApproximate, “greater than” for IsRange, etc. Another challenge with the sub-task is the presence of similar sets of keywords corresponding to multiple Mod types. For example, the Mods ‘IsMean’ and ‘IsMeanHasTolerance’ are very similar with the slight difference that keywords corresponding to the Mod ‘IsMeanHasTolerance’ contain the additional symbol, ‘±’. We adopted a hierarchical approach in order to detect such minute differences and correctly identify the type of Mod for every Q span, e.g. IsMeanHasTolerance is True when IsMean and HasTolerance are both true. We started by detecting a general Mod class, and gradually used extra cues to classify the span into more specific Mod classes such as {IsMeanHasSD, IsMeanHasTolerance, IsRangeHasTolerance, IsList}.

## 4.3 Stage 3

The input to this stage is the sentence-quantity tuple  $\langle s, q \rangle$  and our objective is to detect the spans for ME, MP and Qual. There could be multiple Qs in a single sentence. We treat detecting ME, MP, and Qual as three sequence labeling sub-tasks in a multi-task learning setting.

We create a modified sentence  $s'$  where the Q span  $q$  inside the sentence is enclosed within a special start marker  $\langle E \rangle$ , and a special end marker  $\langle E \rangle$  (Baldini Soares et al., 2019; Kaushal and Vaidhya, 2020; Zong et al., 2020). We additionally have

a special segment embedding for the Quantity ( $q$ ) portion of the quantity-context encoded sentence  $s'$ , different from the remainder of the sentence. We input  $s'$  and corresponding segment embeddings to BERT and obtain quantity-aware contextualized vectors  $\{v_1, v_2 \dots v_n\}$  for each of the  $n$  sub-tokens in  $s'$ . We then obtain the ME task logits  $e_i$ , for each sub-token vector  $v_i$ :

$$e_i = W_e^T v_i + b_e \quad (5)$$

Here  $W_e^T$  and  $b_e$  are learnable parameters. Now, as per the annotation rules of the task, a Q will have an associated MP only if an ME related to the given Q exists. Hence, for predicting the MP, we extract features from the ME task logits and concatenate them with each sub-token vector  $v_i$  as follows:

$$r = [\max_{i=1}^n e_i; \text{mean}_{i=1}^n e_i] \quad (6)$$

$$p_i = W_p^T [v_i; r] + b_p \quad (7)$$

Here  $W_p^T$  and  $b_p$  are learnable parameters,  $\max$  and  $\text{mean}$  are element-wise operations and  $p_i$  is the logit of the  $i^{\text{th}}$  sub-token for the MP sub-task. Here  $;$  denotes concatenation. Similarly, we obtain the logits  $qu_i$  corresponding to the Qual task, for every sub-token vector  $v_i$  of the sentence  $s$ , as follows:

$$qu_i = W_{qu}^T [v_i; r] + b_{qu} \quad (8)$$

Here  $W_{qu}^T$  and  $b_{qu}$  are learnable parameters. The model is trained with the following combined multi-task learning objective:

$$\text{Loss}(s, q, (y_i^e)_{i=1}^n, (y_i^p)_{i=1}^n, (y_i^{qu})_{i=1}^n) = \sum_{j=1}^n (\mathcal{L}(e_j, y_j^e) + \mathcal{L}(p_j, y_j^p) + \mathcal{L}(qu_j, y_j^{qu})) \quad (9)$$

Here  $(y_i^e)_{i=1}^n, (y_i^p)_{i=1}^n, (y_i^{qu})_{i=1}^n$  are ground truths for each sub-token for the ME, MP and Qual sub-tasks respectively;  $\mathcal{L}$  is the softmax cross-entropy loss (Dunne and Campbell, 1997).

Similar to Stage 1, we greedily match the longest contiguous positive labeled spans for each of the three sub-tasks and obtain the ME span  $e$ , MP span  $p$  and Qual span  $qu$  corresponding to the input Q span  $q$  for the sentence  $s$ . Here  $(q, e, p, qu)$  forms an annotation set which is then post processed to generate the relations HP, HQ and QS on this annotation set as per their definitions in §3.

Model	Precision	Recall	F1
BERT-base	0.872	<b>0.972</b>	0.919
BERT-large	0.874	0.933	0.902
RoBERTa-BioMed	0.890	0.951	0.919
SciBERT	<b>0.920</b>	0.889	0.904
BioBERT	0.904	0.946	<b>0.924</b>

Table 1: Stage 1 Results

Model	$F1_{ME}$	$F1_{MP}$	$F1_{Qual}$
BERT Individual	0.499	0.386	0.137
BERT ME, MP	0.515	<b>0.467</b>	N/A
BERT MP, Qual	N/A	0.433	0.166
BERT ME, Qual	0.420	N/A	<b>0.191</b>
BERT ME, MP, Qual	<b>0.517</b>	0.465	<b>0.191</b>
BERT X	0.459	0.330	0.125
BERT Y	0.510	0.428	0.143

Table 2: Multi-Task Results.  $F1_{ME}$ ,  $F1_{MP}$ , and  $F1_{Qual}$  are the F1 measures on the ME, MP, and Qual tasks respectively.

#### 4.4 Domain Specific BERT

Domain specific language model weights lead to a significant performance boost (Müller et al., 2020; Nguyen et al., 2020; Lee et al., 2019; Vaidhya and Kaushal, 2020; Beltagy et al., 2019). SciBERT (Beltagy et al., 2019), BioBERT (Lee et al., 2019), and RoBERTa-BioMed (Liu et al., 2020; Gururangan et al., 2020) performed relatively well as they are pre-trained on scientific documents in domains relevant to our task.

### 5 Experiments and Discussion

All experiments were performed using PyTorch (Paszke et al., 2019) and HuggingFace’s transformers (Wolf et al., 2019). Optimization was done using Adam (Kingma and Ba, 2014). We include the complete set of experimental parameters in §D.

#### 5.1 Development Phase

After dividing the 5 sub-tasks into 3 stages, we worked on each stage individually. We trained the models exclusively on the train dataset and used the trial dataset for validation and hyperparameter tuning. We used the F1, Precision and Recall metrics for each token in the sequence labeling sub-

Model	$F1_{ME}$	$F1_{MP}$	$F1_{Qual}$
BERT-base	0.517	0.465	0.191
BERT-large	0.573	0.446	<b>0.317</b>
RoBERTa-BioMed	<b>0.577</b>	0.473	0.232
SciBERT	0.556	0.486	0.188
BioBERT	0.575	<b>0.501</b>	0.297

Table 3: Stage 3 Results

tasks, for evaluating individual components over the validation set during the development phase.

Table 1 shows the performances of various BERT models in Stage 1. We observe that BioBERT delivers the best F1 score, followed by BERT-base and RoBERTa-BioMed. Much to our surprise, BERT-Large and SciBERT performed worse than BERT-base despite their large size (Li et al., 2020) and domain specificity.

In order to understand the role of each component of our model in Stage 3, we perform various ablation studies as shown in Table 2. First, we experiment with various combinations of multi-task learning with the three tasks - ME, MP and Qual. We observe that multi-task learning can lead to significant gains on all three tasks. Only the multi-task combination of ME and Qual led to performance reduction. Multi-task training all three tasks together nearly gives the best performance on all three metrics. We attribute this gain in performance to the inter-related natures of the three sub-tasks.

Secondly, we study the importance of segmentation and concatenation of features. We create BERT X, which doesn’t add separate segment embeddings for the Q span, and BERT Y which does not concatenate the ME logit features for predicting MP and Qual spans. From Table 2, we observe that BERT X has a significant reduction in performance for all the three sub-tasks upon excluding the segment embeddings, as the model input doesn’t have a clear demarcation between the Q span portion and non-Q span portion of the sentences. We also observe a reduction in performance for MP and Qual for BERT Y, showing the importance of fusing the logits of ME for the former two sub-tasks.

Similar to Stage 1, we experiment with various BERT models as shown in Table 3. Here we observe that RoBERTa-BioMed, BioBERT and BERT-large perform the best for ME, MP and Qual respectively. BERT-Base performs the worst for all of them. All the models except BioBERT have significantly lower  $F1_{Qual}$  than BERT-Large. Each model produces an  $F1_{ME}$  score greater than 0.5.

#### 5.2 Post-Evaluation Phase

The evaluation was done using the official script<sup>3</sup>. The classification and relation extraction sub-tasks were both evaluated by a binary match score and the span identification tasks by a SQuAD style (Ra-

<sup>3</sup><https://github.com/harperco/MeasEval/blob/main/eval>

Model	Q	ME	MP	Qual	Unit	Mod	HQ	HP	QS	Overall
<b>Evaluation Phase</b>										
Baseline	0.815	0.066	0.068	0.028	0.531	0.000	0.081	0.010	<b>0.014</b>	0.225
<b>Submission</b>	0.787	0.113	0.012	0.005	0.748	0.309	0.076	0.006	0.000	0.278
<b>Post-Evaluation Phase</b>										
BERT-base	0.828	0.338	0.277	0.072	0.765	<b>0.465</b>	0.310	0.174	0.000	0.402
BERT-large	0.705	0.343	0.296	0.081	0.755	0.442	0.325	0.207	0.000	0.392
RoBERTa-BioMed	0.812	0.384	0.365	0.104	0.804	0.434	0.383	0.238	0.005	0.440
SciBERT	0.809	0.382	0.324	0.072	<b>0.811</b>	0.435	0.354	0.230	0.000	0.433
BioBERT	<b>0.844</b>	<b>0.407</b>	<b>0.365</b>	<b>0.111</b>	0.796	<b>0.465</b>	<b>0.400</b>	<b>0.269</b>	0.000	<b>0.456</b>

Table 4: Test Set Performance

jpurkar et al., 2016) overlap score. The leaderboard ranking was based on a global F1 score averaged across all sub-tasks.

For our official submission, we selected BioBERT as it achieved the best F1 score in Stage 1 and near-best performance for the tasks in Stage 3. Minor discrepancies in the submission format involving the annot-id reference, quotes, whitespace-sensitivity and utf-8 encoding, not detected by the evaluation script were fixed in the post-evaluation phase. Table 4 shows the final performance of our models. After proper conversion to the desired format during the post-evaluation phase, we also evaluated various other BERT models along with our best model, BioBERT. BioBERT delivers the best performance of 0.456 F1 (Overall) followed by RoBERTa-BioMed and SciBERT. BioBERT also performs best on 7 of the 9 individual tasks.

### 5.3 Future Work

Stage 3 of our pipeline operates at a sentence-level, so for a given Q span, it does not capture the ME, MP, and Qual spans occurring across sentences. However, our approach can be easily extended to consider the nearby sentences or even the entire document (at the cost of computation speed).

The identification of exact word boundaries for the span identification tasks is crucial. Treating these tasks as sequence labeling problems and greedily matching for spans can lead to a few problems. For example, if a sub-token occurring within a long span is mislabeled, then the span is split into two components. In the future, we can explore leveraging contrastive learning (Chen et al., 2020) to improve the predictions for exact word boundary match. We can have transition based labeling layers such as Conditional Random Fields (CRFs) (Wallach, 2004) over the more popular BIO/BIOES sequence tagging schemes (Yang et al., 2018).

Lastly, while the multi-staged approach is fairly interpretable at the intermediate outputs of Q spans,

it also leads to a few issues. The predictions for MP, ME and Qual spans in Stage 3 are heavily dependent on the Q spans from Stage 1, and there does not exist any mechanism to rectify errors in Stage 1 later, in our approach. There is also an exposure bias (Schmidt, 2019; Galloway et al., 2019) as the model is trained on the ground truth, while tested on the predicted Q spans. Moreover, we believe that having common weights between the BERT models of Stage 1 and Stage 3 will not only make our approach faster and lighter, but also more performant through multi-task learning.

## 6 Conclusion

In this paper, we present our system details for the SemEval 2021 Task-8: MeasEval which is aimed at extracting entity and semantic relations pertaining to counts and measurements. We use a multi-staged approach where we first identify the quantity spans using BERT, then the units and modifiers for these predicted quantity spans by intelligent templates that leverage extracted units and modifier keywords. Finally we input the quantity-aware sentences to another BERT model to predict ME, MP, and Qual in a multi-task learning settings with feature re-use. Our submission achieved the second runner up position on the leaderboard for the Unit-identification sub-task and it showed the highest improvement in the post-evaluation phase, with an F1 (Overall) score only 0.063 lower than the highest score across both the phases.

## Acknowledgments

We would like to thank the MeasEval organizers, especially Corey Harper, for their constant support and for clearing our doubts. We would like to thank Aadarsh Sahoo for the insightful discussions. We would also like to thank the Department of Computer Science and Engineering, IIT Kharagpur, for providing us with the computing resources.

## References

- Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. [Matching the blanks: Distributional similarity for relation learning](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [Scibert: Pretrained language model for scientific text](#). In *EMNLP*.
- Soumia Lilia Berrahou, Patrice Buche, Juliette Dibia-Barthelemy, and Mathieu Roche. 2013. How to extract unit of measure in scientific documents? In *KDIR: Knowledge Discovery and Information Retrieval*, pages 454–459. Springer.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Rob A Dunne and Norm A Campbell. 1997. On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function. In *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, volume 181, page 185. Citeseer.
- Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. 2019. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. In *Proceedings of ACL*.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Ayush Kaushal and Tejas Vaidhya. 2020. [Winners at W-NUT 2020 shared task-3: Leveraging event specific and chunk span information for extracting COVID entities from tweets](#). In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 522–529, Online. Association for Computational Linguistics.
- Diederik P Kingma and Jimmy Ba. 2014. [Adam: A method for stochastic optimization](#). *arXiv preprint arXiv:1412.6980*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. [BioBERT: a pre-trained biomedical language representation model for biomedical text mining](#). *Bioinformatics*, 36(4):1234–1240.
- Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joey Gonzalez. 2020. Train big, then compress: Rethinking model size for efficient training and inference of transformers. In *International Conference on Machine Learning*, pages 5958–5968. PMLR.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [RoBERTa: A robustly optimized BERT pretraining approach](#).
- Huaishao Luo, Yu Shi, Ming Gong, Linjun Shou, and Tianrui Li. 2020. [MaP: A matrix-based prediction approach to improve span extraction in machine reading comprehension](#). In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 687–695, Suzhou, China. Association for Computational Linguistics.
- Qiaozhu Mei and Dragomir Radev. 1979. Information retrieval. In *The Oxford Handbook of Computational Linguistics 2nd edition*. Oxford Press.
- Martin Müller, Marcel Salathé, and Per E Kummervold. 2020. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*.
- David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. *arXiv preprint arXiv:2005.10200*.

- Rubaa Panchendrarajan and Aravindh Amaresan. 2018. [Bidirectional LSTM-CRF for named entity recognition](#). In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*, Hong Kong. Association for Computational Linguistics.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. Has-qa: Hierarchical answer spans model for open-domain question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6875–6882.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.
- Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. *EMNLP-IJCNLP 2019*, page 157.
- M Sevenster, J Buurman, P Liu, JF Peters, and PJ Chang. 2015. Natural language processing techniques for extracting and categorizing finding measurements in narrative radiology reports. *Applied clinical informatics*, 6(3):600.
- Mayank Singh, Barnopriyo Barua, Priyank Palod, Manvi Garg, Sidhartha Satapathy, Samuel Bushi, Kumar Ayush, Krishna Sai Rohith, Tulasi Gamidi, Pawan Goyal, and Animesh Mukherjee. 2016. [OCR++: A robust framework for information extraction from scholarly articles](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3390–3400, Osaka, Japan. The COLING 2016 Organizing Committee.
- Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. Brat: a web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107.
- Richard Tzong-Han Tsai, Shih-Hung Wu, Wen-Chi Chou, Yu-Chun Lin, Ding He, Jieh Hsiang, Ting-Yi Sung, and Wen-Lian Hsu. 2006. Various criteria in the evaluation of biomedical named entity recognition. *BMC bioinformatics*, 7(1):1–8.
- Tejas Vaidhya and Ayush Kaushal. 2020. [IITKGP at W-NUT 2020 shared task-1: Domain specific BERT representation for named entity recognition of lab protocol](#). In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 268–272, Online. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Hanna M Wallach. 2004. Conditional random fields: An introduction. *Technical Reports (CIS)*, page 22.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. [Huggingface's transformers: State-of-the-art natural language processing](#). *ArXiv*, abs/1910.03771.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google's neural machine translation system: Bridging the gap between human and machine translation](#). *CoRR*, abs/1609.08144.
- Jie Yang, Shuailong Liang, and Yue Zhang. 2018. Design challenges and misconceptions in neural sequence labeling. *arXiv preprint arXiv:1806.04470*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems*, volume 32.
- Shi Zong, Ashutosh Baheti, Wei Xu, and Alan Ritter. 2020. [Extracting covid-19 events from twitter](#).

Dataset	Total Documents	Q	ME	MP	Qual	Avg. Q	Avg. ME	Avg. MP	Avg. Qual
Train	233	883	875	563	210	3.790	3.755	2.416	0.901
Trial	65	281	273	179	99	4.323	4.200	2.754	1.523
Eval	130	499	499	330	162	3.838	3.838	2.538	1.246

Table 5: Span Statistics. Here *Avg.* signifies the average number of spans present per document.

Dataset	Total Documents	HQ	HP	QS	Avg. HQ	Avg. HP	Avg. QS
Train	233	878	560	210	3.768	2.403	0.901
Trial	65	275	177	99	4.231	2.723	1.523
Eval	130	499	330	162	3.838	2.538	1.246

Table 6: Relation Statistics. Here *Avg.* signifies the average number of relations present per document.

Dependency	Version	Usage
PyTorch	1.4	NN Layers & Autograd
Transformers	4.2	BERT Models
Scikit-learn	0.23	Metrics
SciPy	1.5	Metrics
NLTK	0.5	Sentence Tokenization
Pandas	1.2	Loading Files
Pandasql	0.7	Querying DataFrames
Vladiate	0.0.23	Validating Results
NumPy	1.18	Numerical computation

Table 7: Packages Used

Hyperparameter	Value
Learning Rate	$3e - 5$
Stage 1 Epochs	5
Stage 3 Epochs	10
Batch Size	16
Dropout Final	0.1
BERT Dropout	0.1
Adam: $(\beta, \epsilon)$	$((0.9, 0.999), 1e - 8)$
Weight Decay	0
BERT Configuration	Default
BERT Embeddings	Trainable

Table 8: Best Hyperparameters

## A Appendices

Following is the overview of the appendix.

- §B – We provide implementation details: codebases, trained models and detail dependencies.
- §C – We provide details of the dataset in shared task, its statistics and annotation set for the task.
- §D – We detail the experimental settings and hyperparameters.

## B Code and Dependencies

We will make our code public <sup>4</sup> with instructions to replicate our systems. We also release our pre-

<sup>4</sup><https://github.com/Ayushk4/SE-T8>

trained model for our submissions <sup>5</sup>.

All experiments were performed using PyTorch (Paszke et al., 2019) and HuggingFace’s transformers (Wolf et al., 2019) libraries. The optimization was done using Adam optimizer (Kingma and Ba, 2014). We used git for reproducibility setup. In Table 7 we list all the dependencies used in our codebase. We include a step-by-step guide to setup and run the codebase in our README file present within the code also with details to set up our environment.

## C Dataset Details

We experiment on the dataset provided by the task organizers, consisting of gold annotations (Harper et al., 2021) for the set of scientific documents in English which are released here<sup>6</sup>. These scientific documents are a subset of the Elsevier Labs OA-STM-Corpus available publicly<sup>7</sup>.

**Basic Annotation Set:** The basic annotation set consists of 4 types of spans and 3 types of relations between them. The span types are Quantity (counts and measurements), Measured Entity (the item whose measurement/count is provided by the Quantity spans), Measured Property (the property of the Measured Entity, whose measurement is provided by the Quantity spans) and Qualifier (special circumstances which affect a particular measurement). These spans are related using three types of Relations - HasQuantity (relates a Measured Entity or a Measured Property to a Quantity), HasProperty (relates a Measured Entity to a Measured Property) and Qualifies (relates a Qualifier to any Measured Entity, Measured Property, or Quantity).

<sup>5</sup><https://github.com/Ayushk4/SE-T8/releases>

<sup>6</sup><https://github.com/harperco/MeasEval>

<sup>7</sup><https://github.com/elsevierlabs/OA-STM-Corpus>

Model	Huggingface’s Model API
BERT-base	bert-base-cased
BERT-large	bert-large-cased
RoBERTa-BioMed	allenai/biomed_roberta_base
SciBERT	allenai/scibert_scivocab_cased
BioBERT	dmis-lab/biobert-v1.1

Table 9: BERT Versions

Hyperparameter	Set of Values
Learning Rate	$\{3e - 4, 3e - 5, 3e - 6, 3e - 7\}$
Number of Epochs	$\{5, 10, 15, 20\}$
Batch Size	$\{4, 8, 16, 24\}$

Table 10: Sets of Hyperparameters

**Statistics:** The complete dataset is divided into three parts: train, trial and eval. We train on the train set. Trial is used for validating and Eval is the held-out test dataset on which the final performance of the models are evaluated. In Table 5, we list the dataset statistics for the spans of each type. In Table 6, we list the dataset statistics related to the various relations - (HP, HQ, QS).

## D Experimental Settings

**Preprocessing:** We sentence tokenize every document using the NLTK sentence tokenizer. we observed that phrases such as “Fig. 1”, “Table. 2” and “et al. ”, along with a few others, caused sentences to be tokenized at wrong intervals (due to the presence of “.”). We detected and re-joined the instances for such phrases.

**Normalization:** We normalized the dataset by replacing all numerals by the same digit - 0. The helped our model identify the Q spans better. We observed that without normalization, the F1 (Overlap) Score for Q spans decreased considerably (from 0.844 to 0.790).

**Training and Hyperparameters:** The model take  $\approx 20$  seconds per epoch on Tesla P100. The number of parameters are same as BERT. Table 9 lists the HuggingFace model names corresponding to the BERT models we used. We validated our models using F1 metrics for Stage 1 and Stage 3 over the trial dataset. In Table 10 we share the sets of hyperparameters that we explored whereas in Table 8 we mention the **best** set of hyperparameters that we obtained.



# DPR at SemEval-2021 Task 8: Dynamic Path Reasoning for Measurement Relation Extraction

Amir Pouran Ben Veyseh<sup>1</sup>, Franck Deroncourt<sup>2</sup>,  
and Thien Huu Nguyen<sup>1</sup>

<sup>1</sup> Department of Computer and Information Science, University of Oregon,  
Eugene, OR 97403, USA

<sup>2</sup> Adobe Research, San Jose, CA, USA  
{apouranb, thien}@cs.uoregon.edu,  
franck.deroncourt@adobe.com

## Abstract

Scientific documents are replete with measurements mentioned in various formats and styles. As such, in a document with multiple quantities and measured entities, the task of associating each quantity to its corresponding measured entity is challenging. Thus, it is necessary to have a method to efficiently extract all measurements and attributes related to them. To this end, in this paper, we propose a novel model for the task of measurement relation extraction (MRE) whose goal is to recognize the relation between measured entities, quantities and conditions mentioned in a document. Our model employs a deep translation-based architecture to dynamically induce the important words in the document to classify the relation between a pair of entities. Furthermore, we introduce a novel regularization technique based on Information Bottleneck (IB) to filter out the noisy information from the induced set of important words. Our experiments on the recent SemEval 2021 Task 8 datasets reveal the effectiveness of the proposed model.

## 1 Introduction

One of the key indicators of scientific writing is the quantities description of various experiments and results. While the mentions of all measurements could provide a rigorous understanding of the topic, it might make the reading and automatic processing of the text more difficult. As such, designing effective methods to recognize the mentions of measurements and also the conditions in which they are valid is necessary. According to the definition of the SemEval 2021 Task 8 (Harper et al., 2021), a measurement might consist of the following components: (i) Measure Entity: A span referring to an entity that one of its properties has been measured and its value is provided in the document; (ii) Measured Property: A span referring to the characteristics of an entity that has been measured; (iii)

[ME1] samples [/ME1] have been generated with Coronin and Dystrophin proteins. In the filtration experiments, some of them with a [PR1] diameter[/PR1] [QT1] less than 2 mm [/QT1] have been filtered out using [QT2] 200-degree [/QT2] filtering [ME2] radiation [/ME2], resulting in [QT3] 20% [/QT3] [ME3] utilization [/ME3]. These results are obtained in a [QL1] dry climate [/QL1].

Figure 1: A document annotated with the measured entities (i.e., [ME]), quantity (i.e., [QT]), measured property (i.e., [PR]) and qualifier (i.e., [QL]) (best viewed in color).

Quantity: A span in the document that refers to a value and possibly it comes with a unit; and (iv) Qualifier: A span referring to a condition in which more information about the Quantity, Measured Property or Measured Entity is provided. Figure 1 shows a sample document annotated with the aforementioned entities. In this paper, we collectively name all of these four types as measurement component.

As it is shown in the provided example, documents might contain multiple entities, properties, quantities and qualifiers that are scattered in different parts of the document. As such, finding which measurement components are associated with each other is not straightforward. In this paper, this task is called measurement relation extraction (MRE) that aims to recognize what is the relationship between two given measurement components. More specifically, the following relation types are considered: (i) Has-property: Indicates the selected property is one of the characteristics of the selected entity; (ii) Has-Quantity: Indicates the selected quantity is provided for the selected entity or property; (iii) Qualifies: Indicates the selected qualifier provides more information about the selected entity or quantity; (iv) None: Indicates that there is no

relation between the selected measurement components. For instance, in the given example document in Figure 1, the following relations between different measurement components exist: (1)  $ME_1$  has property  $PR_1$ ; (2)  $PR_1$  has quantity  $QT_1$ ; (3)  $ME_2$  has quantity  $QT_2$ ; (4)  $ME_3$  has quantity  $QT_3$ ; and (5)  $QL_1$  qualifies  $ME_3$ ;

Finding the relation between a pair of measurement components is challenging and it requires consideration about the position of the given entities and the context in which they are used. Generally, this task can be formulated as a typical Relation Extraction (RE) task whose goal is to identify the semantic relation between two given named entity mentions. For RE, it has been shown that contextual information such as dependency path between the two given entities is important. As such, in this paper, we also aim to exploit the contextual information for a pair of measurement entities to predict the relation between them. To this end, the main question to answer is how we can extract the contextual information that is helpful for this task. One simple solution is to use the dependency path between the two measurement components. However, this might not be perfect due to various reasons such as lack of high-quality dependency parser designed especially for scientific domain and the fact that the dependency tree is ignorant of the downstream task (i.e., MRE) thus might not be efficient to extract important context from. Therefore, in this paper, we aim to propose a novel method to dynamically infer the important context for the MRE task. More specifically, we introduce a deep architecture to infer which words should be selected from the given document to form the important context from which the relation between the given measurement components can be inferred. The proposed deep architecture exploits a translation-based perspective to achieve this goal.

In addition, in this paper, we propose a novel method to efficiently regularize the representations of the input words based on the inferred important context. In particular, our method is based on the Information Bottleneck (IB) theory in which the inferred context is treated as information bottleneck to exclude noisy information in the input document representation. We conduct extensive experiments on the SemEval 2021 Task 8 dataset. Our experiments reveal the effectiveness of the proposed model for the task of MRE.

## 2 Model

**Task Definition:** The input to the model is the document  $D = [w_1, w_2, \dots, w_n]$  consisting of  $n$  words and also the positions of the two entities of interest,  $w_s$  and  $w_o$  where  $s$  and  $o$  are the indices of the first (i.e., subject) and the second (i.e., object) entities, respectively. The input document is annotated with the label  $l$  from the set  $L = \{\text{hasQuantity, hasProperty, qualifies, None}\}$ . Our proposed model for this task consists of four major components: (1) Input encoder to convert the input text into high dimensional word vectors; (2) Dependency Path Reasoning: This component employs the word vector representations and extract a path between the two entity mentions in the given document; (3) Regularization: This component employs the extracted dependency path as the information bottleneck to filter out noisy information from the input document; (4) Prediction: Finally the regularized representations of the dependency path will be used to make the final prediction. The rest of this section provides details for the aforementioned components.

### 2.1 Input Encoder

To represent each word  $w_i$  in the input document  $D$ , we use the concatenation of the following components: **Contextualized Embedding**, We feed the input document  $D$ , i.e.,  $[CLS]w_1w_2 \dots w_n[SEP]$  to the pre-trained BERT<sub>base</sub> transformer and take the hidden states of the last layer of the BERT model, i.e.,  $E = [e_1, e_2, \dots, e_n]$ , as the contextualized word embedding of the input document. Note that for the words that have multiple word-pieces, we take the average of their word-piece embeddings obtained from the BERT model. **Position Embedding** For each word  $w_i$ , we compute its distance to the subject  $w_s$  and the object  $w_o$ , i.e.,  $d_s^i = \|i - s\|$  and  $d_o^i = \|i - o\|$ , respectively. The distances are represented using high dimensional vectors  $e_i^s$  and  $e_i^o$  obtained from randomly initialized embedding tables. During training, the embedding tables are being updated. **Entity Type Embedding** The type of the two entities (i.e., Quantity, Measured-Entity, Measured-Property, and Qualifier) are represented using high dimensional vectors obtained from randomly initialized embedding tables. The embedding tables will be fine-tuned during training.

The concatenation of the aforementioned embedding vectors, i.e.,  $X = [x_1, x_2, \dots, x_n]$ , are used

to represent the words of the input document. It is noteworthy that since the parameters of the pre-trained  $BERT_{base}$  are fixed during training, in order to tailor the contextualization of the word embeddings to this task, we feed the vectors  $X$  to a Bi-directional Long Short-Term Memory (BiLSTM) network and we use the hidden states of the BiLSTM neurons, i.e.,  $H = [h_1, h_2, \dots, h_n]$ , as the final vector representations of the input document  $D$ . The vectors  $H$  will be used by the subsequent components.

## 2.2 Dependency Path Reasoning

To find the dependency path between the subject and the object entities, we employ a translation-based perspective. More specifically, given the vector representations of the subject entity, i.e.,  $h_s$ , and the object entity, i.e.,  $h_o$ , the dependency path should be represented using the vector  $P$  such that using this vector, the subject representation  $h_s$  is transferred (i.e., translated) to the object representation  $h_o$ , under the operation  $\Phi$ . Formally,  $h_o = \Phi(h_s, P)$ . Using this definition, we can define the path representation by  $P$  by exploiting the inverse operation  $\Phi^{-1}$ , i.e.,  $P = \Phi^{-1}(h_s, h_o)$ . After obtaining the path representation  $P$ , we compare it with the representations of the other words of the document  $D$  to assess their likelihood to be included in the dependency path. Concretely, the similarity between the vector  $h_i$  and the vector  $P$  could be used to estimate the probability of the word  $i$  to be used in the dependency path. However, one limitation of this method is that the likelihood of the word  $w_i$  is computed regardless of the other words  $w_j$  where  $j \notin \{i, s, o\}$ . To address this issue, we propose to compute the likelihood of the word  $w_i$  based on the interaction between the representation of the word  $w_i$ , i.e.,  $h_i$ , the representations of the other words, i.e.,  $h_j$  for  $j \notin \{i, s, o\}$ , and the path representation  $P$ . To this end, we first compute a vector representation for the words  $w_j$  by applying  $MAX\_POOL$  operation on all words  $w_j$  for  $j \notin \{i, s, o\}$ :  $\bar{h}_{-i} = MAX\_POOL(h_1, h_2, \dots, h_j)$ . Afterwards, we apply the function  $\Phi^{-1}$  on the vectors  $P$  and  $\bar{h}_{-i}$ :  $\hat{h}_i = \Phi^{-1}(\bar{h}_{-i}, P)$ . The vector  $\hat{h}_i$  represents the path for transferring (i.e., translating) the vector  $\bar{h}_{-i}$  to  $P$ . As such, the similarity between  $\hat{h}_i$  and  $h_i$  could reveal how important is the word  $w_i$  to convert the representation of the context  $w_j$  for  $j \notin \{i, s, o\}$  to the representation of the depen-

dency path  $P$ . Therefore, we use this similarity, i.e.,  $Sim_i = \left\| \hat{h}_i - h_i \right\|$ , as the score of the word  $w_i$  to be included in the dependency path. The words that their score is above a pre-defined threshold will be used as the inferred dependency path.

It is worth noting that to learn the function  $\Phi^{-1}$ , in this work, we use a feed forward neural network. In particular, the concatenation of the vectors  $h_s$  and  $h_o$  are fed into a 2-layer feed forward neural network with  $|P|$  neurons at the final layer:  $P = FF([h_s : h_o])$ , where  $[:]$  represents concatenation and  $FF$  represents the feed-forward neural network. To train the  $FF$  network for the RE task, we use the vector  $P$  to predict the probability distribution  $P_\Phi(\cdot|D, t, a)$  using another feed-forward network  $FF_2$  whose final layer dimension equals the number of labels, i.e.,  $|L|$ . We use negative log-likelihood to train the  $FF$  and  $FF_2$  networks:  $\mathcal{L}_\Phi = -\log(P_\Phi(l|D, t, a))$  where  $l$  is the gold label.

Finally, to represent the induced path, we take the max-pooled representation of the words in the path:  $h_P = MAX\_POOL(h_1, h_2, \dots, h_p)$  where  $p$  is the number of words in the induced dependency path. The path representation  $h_p$  will be used by the subsequent components.

## 2.3 Regularization

Although the induced dependency path from the previous component is intended to contain the important information for the RE task, it might still contain some noisy information due to the contextualization in the input encoder. To overcome this noisy information, in this work, we propose to exploit the induced path as the information bottleneck (IB) (Tishby et al., 2000). IB’s goal is to reduce the mutual information between the input and the bottleneck, meanwhile, to increase the mutual information between the bottleneck and the output. For the second goal, the bottleneck (i.e., the dependency path representation  $h_p$ ) will be used by the prediction component, and the increase of its mutual information with the output is enforced by reducing the training loss (e.g., negative log-likelihood). To fulfill the first goal, i.e., decreasing the mutual information between the input and the bottleneck, we resort to a contrastive learning paradigm to estimate the mutual information between two high-dimensional vectors using the classification loss of a binary-discriminator. More specifically, the path representation  $h_p$  is concatenated with the

max-pooled representation of the input document  $D$ , i.e.,  $h_d = \text{MAX\_POOL}(h_1, h_2, \dots, h_n)$ , and this concatenation, i.e.,  $h_{pos} = [h_p : h_d]$ , serves as the positive sample for the contrastive learning. To construct the negative samples, we first take the max-pooled representation of a randomly chosen document  $D'$  from the same mini-batch, i.e.,  $h_{d'} = \text{MAX\_POOL}(h'_1, h'_2, \dots, h'_m)$  where  $h'_i$  is the representation of the  $i$ -th word in the document  $D'$  and  $m$  is the total number of words in  $D'$ . Afterwards, the concatenation of  $h_p$  and  $h_{d'}$  is employed as the negative sample:  $h_{neg} = [h_p : h_{d'}]$ . Finally, a feed-forward discriminator is employed and trained to distinguish the positive samples from the negative ones, i.e.,  $\mathcal{L}_{disc} = \log(1 + e^{(1-D(h_{pos}))}) + \log(1 + e^{D(h_{neg})})$ . By adding the discriminator loss  $\mathcal{L}_{disc}$  to the final loss function and decreasing it, the estimated mutual information between the input and the bottleneck (i.e., the path representation  $h_p$ ) is decreased too.

## 2.4 Prediction

To make the final prediction on the relation between the given subject and object entities, we employ the representations of the induced dependency path (i.e.,  $h_p$ ), the subject entity (i.e.,  $h_s$ ), and the object entity (i.e.,  $h_o$ ) to construct the final vector  $V = [h_p : h_s : h_o]$  where  $[:]$  represent concatenation. The vector  $V$  is finally consumed by a feed-forward neural network to predict the distribution  $P(\cdot|D, t, a)$ . The loss function to train the main RE task is thus defined as:  $\mathcal{L}_{pred} = -\log(P(l|D, t, a))$  where  $l$  is the gold label. The overall loss function to train the entire model is:  $\mathcal{L} = \mathcal{L}_{pred} + \alpha\mathcal{L}_\Phi + \beta\mathcal{L}_{disc}$  where  $\alpha$  and  $\beta$  are the trade-off parameters.

## 3 Experiments

### 3.1 Dataset, Hyper-Parameters & Baselines

In order to demonstrate the effectiveness of the proposed model, i.e., Dynamic Path Reasoning (DPR), we evaluate it on the recent SemEval 2021 Task 8 dataset. This dataset provides measurement annotation for 233 training documents, 65 development documents, and 130 testing documents, all in English. Note that we do experiments only on the train and trial set (as the gold entities are not available for test set). Also, we evaluate the model only for relation extraction, not the entire task (as such, we did not make a submission during MeasEval evalu-

ation phase). More specifically, for each document, the positions of the measured entities, measured properties, quantities, and qualifiers are provided. Furthermore, for each measurement component, its relations with the other components or extra information (e.g., unit of quantity) is available. Note that in our experiments, we do not use the *annotation set* information which indicates which components belong to the same measurement.

We fine-tune the hyper-parameters of the proposed model on the development set of the SemEval 2021 Task 8 dataset. The model with the best performance on the development set is evaluated on the test set. Based on our experiments, the following hyper-parameters are selected: 50 dimensions for the position embedding and entity type embedding; 200 dimensions for the hidden layer of the BiLSTM and all feed-forward networks; 0.1 and 0.05 for the trade-off parameters  $\alpha$  and  $\beta$ ; 0.7 for the threshold in the dynamic path reasoning component; Adam optimizer with learning rate 0.3; batch-size 50; and early stopping with the patience of 10.

To comprehensively evaluate the proposed model, we compare its performance against the following baselines: (i) Sequential Models, specifically we compare with **BiLSTM** which takes the non-contextualized word embeddings of the input document (i.e., GloVe) and encode the sequence of the words. Moreover, we also compare with **BERT** model fine-tuned during training for the MRE task. (ii) Structure-aware models, these models employ the structure of the input document (e.g., dependency trees of the sentences). Specifically, we compare with iDepNN (Gupta et al., 2019) which employs the dependency trees of the sentences of the document. This baseline adds an edge between the roots of the trees to create a connected graph, Furthermore, it prunes the tree along the dependency path between the two entities of interest. Finally, we compare our model with **LSR** which dynamically infer a graph structure for the input document using the representations of the entities and other words on the dependency path between the entities.

### 3.2 Results

The results on the test set are presented in Table 1. There are several observations from this table. First, the proposed model significantly (with  $p < 0.01$ ) outperforms the baselines. It indicates the importance of using dynamic path reasoning and also the

Model	Precision	Recall	F1
<b>BiLSTM</b>	65.3	71.1	68.1
<b>BERT</b>	70.4	71.8	71.1
<b>iDepNN</b>	69.4	75.0	72.4
<b>LSR</b>	72	75.9	73.9
<b>DPR (Ours)</b>	70.1	83.4	<b>76.2</b>

Table 1: Performance on Test set

proposed regularization method. Second, Comparing the structure-aware and sequence-based baselines, it is evident that the structure of the input document is necessary for achieving better results. However, between the iDepNN and the LSR baseline, the latter has better performance due to its capability of inferring the structure of the document instead of relying on external parse trees as in iDepNN. Finally, this experiment shows that using the pre-trained language model BERT substantially improves the performance compared to a sequence-based model that utilizes GloVe embedding. This is on par with the recent advancement on NLP using contextualized word embeddings.

### 3.3 Ablation Study

In this section, we provide more insight into the effectiveness of different components of the proposed model. The major two components in our model are dynamic path reasoning and regularization. To study their importance, we evaluate the performance of the following baselines on the development set of the SemEval 2021 Task 8 datasets: (i) **Full<sup>-DPR</sup>**, this baseline completely removes the dynamic path reasoning component. More specifically, the vector  $h_p$  is removed from the final prediction vector  $V$  and the loss function  $\mathcal{L}_\Phi$  is also removed from the overall loss function  $\mathcal{L}$ ; (ii) **Full<sup>DPRS</sup>**, this baseline employs the dynamic path reasoning component. However, to compute the similarity score  $Sim_i$ , instead of considering the context if the word  $w_i$ , it directly computes the score by  $Sim_i = \|P - h_i\|$ ; (iii) **Full<sup>-Reg</sup>**, this model completely remove the regularization component, i.e., by removing the loss function  $\mathcal{L}_{disc}$  from the overall loss function  $\mathcal{L}$ ; (iv) **Full<sup>dot</sup>**, this ablated model preserves the regularization component. However, instead of using Information Bottleneck, it directly decreases the similarity between the path representation, i.e.,  $h_p$ , and the input document representation, i.e.,  $h_d$ , by replacing the  $\mathcal{L}_{disc}$  by  $\mathcal{L}_{\lceil \sqcup} = h_p \cdot h_d$ .

The results are presented in Table 2. This table shows that all components of the proposed model

Model	Precision	Recall	F1
<b>Full</b>	73.2	86.7	79.4
<b>Full<sup>-DPR</sup></b>	71.1	79.1	74.9
<b>Full<sup>DPRS</sup></b>	70.2	82.3	75.8
<b>Full<sup>-Reg</sup></b>	72.9	80.6	76.6
<b>Full<sup>dot</sup></b>	73.8	76.4	75.1

Table 2: Performance of the ablated models on the development set

are necessary to achieve the highest performance. More specifically, the dynamic path reasoning has the highest impact on the performance as removing it will hurt the most. Also, it shows that the consideration of the context to compute the score for each word to be included in the induced path is necessary. Finally, it shows that regularization is helpful for exclude noisy information from the input. More interestingly, replacing the IB with a dot product to enforce the regularization hurts more than removing the regularization itself. It indicates the necessity of using IB for regularization.

## 4 Related Work

Measurement Relation Extraction (MRE) is one specific formulation of the general Relation Extraction (RE) task. In the literature, RE has been tackled by feature-based methods (Zelenko et al., 2003; Zhou et al., 2005; Sun et al., 2011; Nguyen and Grishman, 2014; Nguyen et al., 2015c) and advanced deep learning models (Zeng et al., 2014; Wang et al., 2016; Lee et al., 2017; Zhang et al., 2017; Nguyen et al., 2019; Jin et al., 2018; Veyseh et al., 2020b). Recently, structure-aware deep models have shown significant improvement for RE (Peng et al., 2017; Song et al., 2018; Xu et al., 2015; Liu et al., 2015; Miwa and Bansal, 2016; Nguyen and Grishman, 2018a; Zhang et al., 2018). For a thorough review of the prior works, refer to the recent work (Gupta et al., 2019; Nan et al., 2020; Veyseh et al., 2020a)

## 5 Conclusion

We proposed a new model for the MRE task. The introduced model employs a dynamic path reasoning component which induces important context words to predict the relation between two measurement components. Furthermore, we proposed a novel regularization method based on Information Bottleneck to exclude noisy information from the input. Our experiments on the SemEval 2021 Task 8 reveal the effectiveness of the proposed model.

## Acknowledgments

This research has been supported by the Army Research Office (ARO) grant W911NF-21-1-0112. This research is also based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

## References

- Pankaj Gupta, Subburam Rajaram, Hinrich Schütze, and Thomas Runkler. 2019. Neural relation extraction within and across sentence boundaries. In *AAAI*.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Di Jin, Franck Deroncourt, Elena Sergeeva, Matthew McDermott, and Geeticka Chauhan. 2018. MIT-MEDG at SemEval-2018 task 7: Semantic relation classification via convolution neural network. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 798–804.
- Ji Young Lee, Franck Deroncourt, and Peter Szolovits. 2017. MIT at SemEval-2017 task 10: Relation extraction with convolutional neural networks. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 978–984.
- Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *ACL*.
- Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In *ACL*.
- Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *ACL*.
- Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *ACL*.
- Thien Huu Nguyen and Ralph Grishman. 2018a. Graph convolutional networks with argument-aware pooling for event detection. In *AAAI*.
- Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015c. Semantic representations for domain adaptation: A case study on the tree kernel-based method for relation extraction. In *ACL-IJCNLP*.
- Tuan Ngo Nguyen, Franck Deroncourt, and Thien Huu Nguyen. 2019. On the effectiveness of the pooling methods for biomedical relation extraction with deep learning. *arXiv preprint arXiv:1911.01055*.
- Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph lstms. In *TACL*.
- Linfeng Song, Yue Zhang, Zhiguo Wang, and Daniel Gildea. 2018. N-ary relation extraction using graph state lstm. In *EMNLP*.
- Ang Sun, Ralph Grishman, and Satoshi Sekine. 2011. Semi-supervised relation extraction with large-scale word clustering. In *ACL*.
- Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. In *arXiv preprint physics/0004057*.
- Amir Veyseh, Franck Deroncourt, My Thai, Dejing Dou, and Thien Nguyen. 2020a. Multi-view consistency for relation extraction via mutual information and structure prediction. In *AAAI*.
- Amir Pouran Ben Veyseh, Franck Deroncourt, Dejing Dou, and Thien Huu Nguyen. 2020b. Exploiting the syntax-model consistency for neural relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8021–8032.
- Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *EMNLP*.
- Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*.
- Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. In *Journal of machine learning research*.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*.

Yuhao Zhang, Peng Qi, and Christopher D Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *EMNLP*.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D Manning. 2017. Position-aware attention and supervised data improve slot filling. In *EMNLP*.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL*.

# CLaC-np at SemEval-2021 Task 8: Dependency DGCNN

Nihatha Lathiff, Pavel Khloponin, and Sabine Bergler

CLaC Labs, Concordia University

Montreal, Canada

{f.lathiff, p.khlopo, bergler}@cse.concordia.ca

## Abstract

MeasEval aims at identifying quantities along with the entities that are measured with additional properties within English scientific documents. The variety of styles used makes measurements, a most crucial aspect of scientific writing, challenging to extract. This paper presents ablation studies making the case for several preprocessing steps such as specialized tokenization rules. For linguistic structure, we encode dependency trees in a Deep Graph Convolution Network (DGCNN) for multi-task classification.

## 1 Introduction

Scientific articles contain many quantities which have to be linked to their measured entities. Identifying quantities may seem as simple as digit recognition, but numbers alone are not informative. The entities and properties being measured, while crucial information, are difficult to extract. SemEval 2021 Task 8 (Harper et al., 2021) is a semantic relation extraction task consisting of 5 subtasks: identifying *quantities* and their *modifying* attributes, identifying *measured entities* and their *properties* as well as *qualifying* attributes, if specified.

Recent reports on the strong performance of purely neural models for NLP tasks often underreport the data preprocessing and postprocessing steps that accompany them. Preprocessing significantly influences overall performance. Typical NLP preprocessing steps include sentence splitting and tokenization, sometimes followed by task relevant gazetteer annotation, possibly named entity recognition (NER), part-of-speech (POS) tagging and dependency parsing. These preprocessing steps are so common that many different packages perform them, such as Stanford CoreNLP (Manning et al., 2014), spaCy<sup>1</sup>, and NLTK (Bird et al., 2009).

<sup>1</sup><https://spacy.io/models>

Linguistically inspired features are, however, not regularly exploited and we present here one attempt at encoding dependency information for the structural task of linking quantities with their measured entities, measured properties or qualifiers.

We approach the MeasEval task as a multi-class classification task using a Deep Graph Convolution Neural Network (DGCNN) (Zhang et al., 2018), treating the dependency parse tree as a graph to convolve over. We explore tokenization variants, as well as encodings of the dependency relations using node2vec (Grover and Leskovec, 2016) and UMAP (McInnes et al., 2018) techniques.

## 2 Problem Statement

The MeasEval (Harper et al., 2021) task consists of 5 (not independent) sub-tasks covering span detection, classification and relation extraction across multiple sentences<sup>2</sup>. Given a paragraph of scientific content in English, a system should: 1) label quantity spans (Q) where Q can be simple count or a numerical value with a unit. 2) if there is a unit it should be labelled as Unit(U), and a Q should be classified into one of the types (*count*, *approximate*, *range*, *list*, *mean*, *median*, *medianHasSD*, *meanHasTolerance*, *rangeHasTolerance*, *hasTolerance*) as Modifier(mod). 3) for each Q, systems should identify the span of a measured entity (ME) if one exists and also any measured properties (MP). 4) Identify any spans of qualifiers (QL) that record additional detail related to Q, ME or MP. 5) Label the relationships between Q, ME, MP and QL spans using HasQuantity(HQ), HasProperty(HP) and Qualifies relation types.

## 3 System Overview

**Motivation:** Unlike named entity detection tasks, MeasEval’s ME or MP detection depends on quan-

<sup>2</sup><https://competitions.codalab.org/competitions/25770>



ties and their relation to other tokens within a sentence. Since dependency parse trees are capable of providing approximations of semantic relationships between predicates and their arguments, we opted to generalize over different dependency parse trees to obtain latent path representations to distinguish between different semantic connections that quantities have with MEs or MPs. To encode this path representation we use a Graph Convolutional Networks (GCN) (Kipf and Welling, 2017) in the form of a DGCNN (Zhang et al., 2018), to operate directly on the dependency graph to capture higher order neighborhood information in the form of embeddings. This embedding is used for classifying the relationship type between the tokens to detect one of the 6 classes explained in Section 3.2.2

Our system has 3 main phases: preprocessing, input creation and training a GCN model and post processing respectively. Each phase communicates with the next through CoNLL format (Tjong Kim Sang and De Meulder, 2003) files.

### 3.1 Phase-I: Preprocessing

We preprocess data using the GATE (Cunningham et al., 2013) modules: ANNIE Tokenizer, ANNIE Sentence splitter, and Stanford Parser (POS tags and dependency graphs (de Marneffe et al., 2006)). Special tokenization rules added are:

**mixed character protection** prevents splitting tokens of differing character types into different tokens, e.g.  $\delta 13CTOC \not\rightarrow \delta, 13, CTOC$

**split mathematical symbols** preserves the usual ANNIE tokenization for  $5 \leq 2\theta/^\circ \leq 80$  into seven tokens ( $5, \leq, 2\theta, /, ^\circ, \leq, 80$ )

**number normalization** decimal numbers are prevented from being split into 3 tokens. Number words are also identified as numbers

**abbreviation period** common abbreviations in scientific journals are recognized as integral tokens including the abbreviation period, e.g. *e.g., Fig., sp., spp.* This improves sentence splitting and tokenization

**list and interval protection** scientific articles frequently report on intervals expressed in different ways and on lists of variable lengths. Both do not usually receive proper parse assignments, because the group as a whole plays a role in the text. To improve the dependency

relation assignments, we manually assign the POS tag ‘CD’ to the groupings:

CD (: | - | to) CD  
 CD (, CD)\* and CD

**unit gazetteer** composed from different sources<sup>3</sup> listing 4280 units

### 3.2 Phase-II: Input creation and GCN training

We train a DGCNN (Zhang et al., 2018) as a multilayer neural network that operates directly on a graph to induce node embeddings with properties of their neighborhood. DGCNN takes  $(A, I)$  as input, where  $A \in \mathbb{R}^{n \times n}$  is an adjacency matrix and  $n$  is equal to the number of nodes in the graph  $G$ .  $I \in \mathbb{R}^{n \times c}$  is an information matrix, associating  $c$  feature values with each of the  $n$  nodes. A single layer of DGCNN captures information according to:

$$Z = f(\hat{D}^{-1}AIW) \quad (1)$$

where  $\hat{D}$  is a diagonal degree matrix with  $\hat{D}_{ii} = \sum_j A_{ij}$  (capturing the branching factor of node  $i$ ) and  $W \in \mathbb{R}^{c \times c'}$  is a trainable parameter matrix.  $f$  is a nonlinear activation function and  $Z \in \mathbb{R}^{n \times c'}$ . Higher order neighborhood information is obtained by stacking multiple DGCNN layers:

$$Z^{t+1} = f(\hat{D}^{-1}AZ^tW^t) \quad (2)$$

where  $Z^0 = I$ ,  $Z^t \in \mathbb{R}^{n \times c_t}$  is the output of the  $t^{th}$  graph convolution layer,  $c_t$  is the size of the output vector of layer  $t$  and  $W^t \in \mathbb{R}^{c_t \times c_{t+1}}$ .

We model a dependency tree as graph  $G = (V, E)$ , where  $V$  are tokens and  $E$  are directed dependency relations. We ensure  $(v, v) \in E$  for all  $v \in V$ . To represent paths in the dependency graph between any two nodes, we add explicit reverse links (to *nsubj* from governor to dependant we add *rnsbj* from dependant to governor).

#### 3.2.1 Input creation

The DCNN classifier predicts six output classes, as defined in Section 3.2.2 for token pairs  $(t_1, t_2)$ , the *subgraph center points*.

The following sections show how (i) candidate token pairs are created, (ii) the smallest subgraph containing  $t_1$  and  $t_2$  is extracted, (iii) each subgraph  $SG$  is represented by  $(A_{SG}, I_{SG})$

<sup>3</sup><http://www.ibiblio.org/units/index.html>, [https://en.wikipedia.org/wiki/Metric\\_units](https://en.wikipedia.org/wiki/Metric_units)

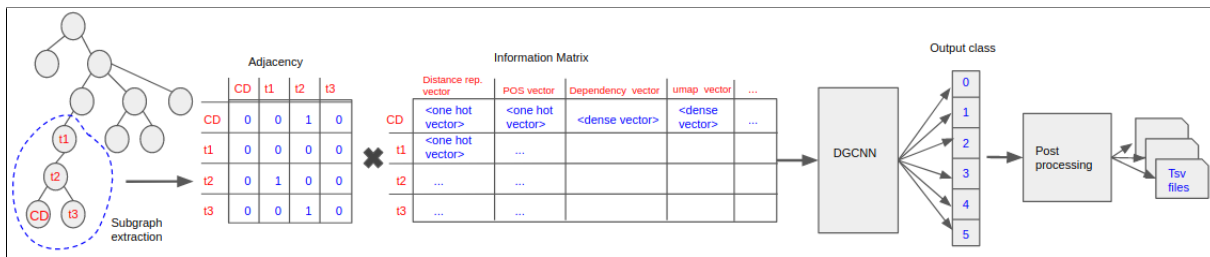


Figure 1: System Architecture from phase-II to phase-III

**Subgraph center point candidates:** In all pairs  $(t_1, t_2)$ ,  $t_1$  has to be a CD (a candidate for a quantity).<sup>4</sup>

In principle, all nodes in the graph are candidates for  $t_2$ , but we empirically set a limit of five on the connecting path length.

**Subgraph extraction** For each  $(t_1, t_2)$ , we select a subgraph containing only the shortest path recursively:  $SG_1$  contains all neighbors of  $t_1$ .  $SG_{k+1}$  contains  $SG_k$  and all neighbors of  $SG_k$ . We select the first subgraph that contains  $t_2$ .

**Adjacency matrix representation** The adjacency matrix  $A \in \mathbb{R}^{n \times n}$  is a binary matrix. Dependency relations from governor to dependant are one-to-many and all rows in the matrix are interpreted as dependants, the columns as governors.

**Information matrix** The information matrix  $I$  is a  $n \times c$  matrix, where  $c$  is size of the concatenated values for the five explicit or latent features associated with each vertex in our system:

**DISTANCE FEATURE** We use the Double Radius Node Label (DRNL) (Zhang and Chen, 2018) to calculate a combined distance of a node  $v_i$  to both subgraph centre nodes within the information matrix as one hot vector as follows:

Nodes  $t_1$  and  $t_2$  carry the label 1. For each  $v$  in the subgraph we calculate labels representing distance using the following hashing function:

$$f(v) = 1 + \min(d_{t_1}, d_{t_2}) + \lfloor \frac{d}{2} \rfloor \times (\lfloor \frac{d}{2} \rfloor + [d\%2] - 1) \quad (3)$$

where  $f(v)$  assigns labels to all nodes  $v$ ,  $d_{t_1}$  and  $d_{t_2}$  are distances of  $v$  with respect to  $t_1$  and  $t_2$  respectively.  $d = d_{t_1} + d_{t_2}$ ,  $\lfloor d/2 \rfloor$  is the integer quotient and  $[d\%2]$  is the remainder of  $d$  divided by 2.

<sup>4</sup>Note that the gold relations connect text spans, not necessarily tokens. The classifier attempts to predict relations between tokens and the postprocessing phase maps the results to spans.

POS FEATURE encoded as a one hot vector

WORD EMBEDDING from the PubMed ELMo model (Peters et al., 2018) of size 1024.

DEPENDENCY PATH EMBEDDING represents the dependency path of each node within the subgraph from  $t_1$  (base node)  $(p(v, t_1))$ . We create *dependency embeddings* of size 128 from dependency sequences in the training data using node2vec (Grover and Leskovec, 2016). Given a graph  $G$ , node2vec<sup>5</sup> uses a random walk procedure from each node  $v \in G$  to produce  $s$  sequences of length  $l$ , and uses these sequences for training node2vec. We embed dependency sequences instead of node sequences to produce embeddings for each dependency relationship type (i.e. we use node2vec to produce embeddings for edges instead of nodes). To represent  $p(v, t_1)$  we concatenate dependency embeddings for each dependency along the dependency path. Given our empirical limit,  $p(v, t_1) \in \mathbb{R}^{5 \times 128}$  and for smaller subgraphs we pad with 0's.

UMAP EMBEDDING As either a complement, or a replacement to dependency embeddings, we experimented with the UMAP dimension reduction technique (McInnes et al., 2018). We trained UMAP as a supervised learning approach feeding dependency embeddings along with class labels and reduced dependency path embeddings to 2 dimensions.

### 3.2.2 Training the DGCNN model

Input  $(A, I)$  was used to train an off the shelf implementation of Deep Graph Convolution Neural Networks (DGCNN)<sup>6</sup> with CrossEntropyLoss<sup>7</sup> as its loss function and with class weights calculated

<sup>5</sup><https://github.com/aditya-grover/node2vec>

<sup>6</sup>[https://github.com/muhanzhang/pytorch\\_DGCNN](https://github.com/muhanzhang/pytorch_DGCNN)

<sup>7</sup><https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>

by  $1/\text{total\_num\_datapoints\_within\_class}$ . We trained the system for 6 epochs with a batch size of 100 in a cuda environment with 6 output classes to predict.

**Class labelling:** Center points  $(t_1, t_2)$  are predicted to fall into one of six classes:

**Class 0:**  $t_1$  is not part of a gold Quantity (Q) span<sup>8</sup>; **Class 1:**  $t_1$  is within a Q span but  $t_2$  is not within any gold span; **Class 2:**  $t_1$  and  $t_2$  are in the same Q span (e.g. 5 kg); **Class 3:**  $t_1$  is within a Q span and  $t_2$  is any token within the ME span belonging to the same annotation set as  $t_1$ ; **Class 4:**  $t_1$  is within a Q span and  $t_2$  is any token within the MP span belonging to the same annotation set as  $t_1$ ; **Class 5:**  $t_1$  is within a Q span and  $t_2$  is any token within the QL span belonging to the same annotation set as  $t_1$ .

### 3.3 Phase-III: Postprocessing

The six classifier classes predict relations between two tokens, while the gold standard annotates relations between text spans. The required mapping to competition output requires several postprocessing steps.

**Prediction ranking** When multiple  $t_2$  with the same class are predicted for a particular  $t_1$  we choose  $t_2$  with the highest probability.<sup>9</sup>

**Span mapping** For each prediction  $(t_1, t_2) \in \text{ClassY}$  we record the token offset for  $t_2$  as span for ClassY in our system output unless the BERT model from (Therien et al., 2021) finds a neighbouring token with the same predicted class, in which case they are merged into the same span.

**Detecting units** Once quantity spans are determined, any measurement gazetteer entry within it was labelled as a unit.

**Predicting quantity modifiers** We used another pretrained BERT model from (Therien et al., 2021) to predict modifiers for each quantity span.

## 4 Results and Analysis

We split the combined training and trial datasets randomly into 75% training and 25% validation set resulting in 233 and 80 documents in our training and validation set respectively.

<sup>8</sup>Not all CDs are part of a gold quantity (e.g. Fig. 5).

<sup>9</sup>We exclude predictions for  $(t_1, t_2)$  where  $t_2$  carries POS tags among *IN*, *DT*, *CD*, *PUNCT*.

We experimented with different preprocessing and feature representations using the official Meas-Eval evaluation script. DGCNN training parameters were fixed for all the experiments.

Table 1 shows development results, the first row (in italics) shows the competition system. The first column (T) indicates the influence of the list and interval protection step: *c* indicates it is included, *n* indicates it is not included. We observe that not including it returns slightly better results, offset by a high rate of duplicates<sup>10</sup>.

We experiment with different path length limits (column 4: H). While a length limit of 8 showed better results on the development data than our competition limit of 5, the same is not true for the test data (see Table 2), where there is no equivalent recall gain.

T:	S:	H:	P	R	F1	EM	O	
<i>c</i>	<i>s</i>	<i>u</i>	5	.586	.466	.520	.281	.328
<i>c</i>	<i>s</i>	<i>u</i>	8	.563	.503	.532	.288	.336
<i>n</i>	<i>s</i>	<i>u</i>	5	.567	.485	.522	.291	.334
<i>n</i>	<i>s</i>	<i>u</i>	6	.591	.487	.535	.301	.344
<i>n</i>	<i>s</i>	<i>u</i>	8	<b>.654</b>	<b>.529</b>	<b>.585</b>	<b>.337</b>	<b>.387</b>

Table 1: Development results. P:precision, R:recall, F1:F1-score, EM:exact match, O:F1 (Overlap)

Table 2 shows competition and post-competition results on test data. The initial competition system is in italics, in bold are the revised results after the organizers removed duplicates.

T:	S:	H:	P	R	F1	EM	O	
<i>c</i>	<i>s</i>	<i>u</i>	5	<b>.546</b>	<b>.323</b>	<b>.406</b>	<b>.217</b>	<b>.241</b>
<i>c</i>	<i>s</i>	<i>u</i>	5	.554	.347	.427	.232	.258

Table 2: Competition results

Results on the test data are significantly lower, indicating overfitting. Post-competition ablation in Table 3 shows that UMAP, for instance, was not effective. Overlap determined competition rankings.

T:	S:	H:	P	R	F1	EM	O	
<i>c</i>	<i>s</i>	<i>u</i>	5	.691	.313	.431	.241	.264
<i>c</i>	<i>s</i>	<i>u</i>	5	.539	.446	.448	.275	.306
<i>c</i>	<i>s</i>	<i>u</i>	8	.614	.310	.412	.227	.249
<i>c</i>	<i>s</i>	<i>u</i>	8	.480	.477	.478	.263	.296

Table 3: Post competition results

<sup>10</sup>Only in the first row of Table 2 are duplicates removed, all other reported results have duplicates and thus overreport slightly.

	Q	ME	MP	U	Mod	HQ	HP
csu5	<b>.889</b>	<b>.057</b>	<b>.007</b>	<b>.495</b>	<b>.408</b>	<b>.028</b>	<b>0.</b>
csu5	.773	.063	.007	.485	.432	.028	0.
csu5	.778	.006	0.	.431	.449	.004	0.
cs_5	.830	.177	.177	.449	.484	.167	.053

Table 4: F1 overlap scores of annotation types on competition test data. Q:Quantity, ME:Measured Entity, MP: Measured Property, U:Unit, Mod:Modifier, HQ: Has Quantity, HP: Has Property

Analysing performance for the different labels in Table 4 shows that our system is not yet mature and needs adjusting. The potential of the DCGNN for the tasks is demonstrated by the high results for quantity (Q) and acceptable results for units (U), which are in line with stronger systems. The comparatively low performance for measured entities (ME) and measured properties (MP) demonstrates that the multi-class labelling approach needs better support. We will consider approaches from the literature (Yao et al., 2018), (Sun et al., 2019), (Hong et al., 2020), (Gupta et al., 2016).

HasQuantity and HasProperty received no attention during development and consistently scored 0 for runs with UMAP, see Table 4.

**Combined tokens:** When  $CD(CD),*$  and  $CD$  is followed by *respectively*, each list item corresponds to a different entity/property (e.g. *This compares to signatures of accelerated electron precipitation from peaked electrons and “inverted-V” electrons, which occur on 9.8 and 3.4% of MEX orbits, respectively.*) As per gold annotation, 9.8 and 3.4% are 2 different Qs both with “MEX Orbits” as ME and “signatures of accelerated electron precipitation” as MPs. We generate *9.8 and 3.4%* as a single Q and “Signatures” as ME, leading to several false negatives. Documents S0032063312003054-2458, S0016236113008041-3257, S0378112713005288-1916 caused this error.

**Math equalities** Our system does not protect math environments such as *...tetragonal unit cell with  $a=4.1816(4)$  Å and  $c=10.0322(6)$  Å...* in document S0022459611006116-1351. Consequently,  $a$  is labelled as a determiner (DT). As determiners are not part of gold labels,  $a$  is eliminated in postprocessing when it should have been labelled as an MP in this case and also in document S0022459611006116-1257.

**Calculations with measurements** Our custom tokenizer does not handle calculations, as in “...re-

*vised average base reaction rate (from  $k_1 = 2 \times 10^{-9}$  to  $k_1 = 1 \times 10^{-9} \text{ cm}^3 \text{ s}^{-1}$ )...*” in document S0019103512003533-3908. The gold annotation for Q is ( $k_1 = 2 \times 10^{-9} \text{ to } k_1 = 1 \times 10^{-9} \text{ cm}^3 \text{ s}^{-1}$ ) and ME is *average base reaction rate*, where we return “ $2 \times 10$ ” as a Q with “average” as ME, “ $1 \times 10$ ” as Q with “reaction” as ME and “9” (split by symbol “-”) as separate quantities with ME as “reaction”, incurring false positive errors.

**Duplication of Quantities** Lists of quantities including units are not combined with our custom tokenizer, thus for *for 4.5 kg and 6 kg samples* in document S0016236113008041-3257, we stipulate two different quantities. This results in duplication of quantities in our submission.<sup>11</sup>

**Units** The gold standard annotates for instance *thin shale barriers* (S1750583613004192-1126), *das* (S037842901300244X-1654),  $\% \Delta E/E$  (S0301010413004096-767), and *KLoC* (S016412121300188X-3207) as units. These are not included in our gazetteer list of 4028 units, incurring false negative errors.

## 5 Conclusions

The MeasEval task is a challenging task that can benefit from a variety of tools in a well integrated system. The small data size limits a true appreciation of the challenges involved but ablation studies suggest that tokenization variations influence precision and recall differently and should be carefully considered in application systems. Also, a umap reduction suggested a ca 1% performance boost on validation data, but incurs a ca 5% loss on test data, showing signs of overfitting. The subgraph classifier proved effective only for quantity and unit prediction. Ablations show that larger subgraphs and longer paths lead to performance degradation, making a case for more task oriented locality features. Duplicates have to be removed.

While the unusual complexity of the classification task and the limited size of the dataset prohibits very general conclusions, we showed that DCGNNs offer an interesting way to encode dependency information but that it has to be supported by several domain inspired contributions to work for all task components effectively.

<sup>11</sup>This duplication of labels was removed by the organizers for the official ranking.

## 6 Acknowledgments

We gratefully acknowledge Benjamin Thérien and Parsa Bagherzadeh for their help. The work was supported by NSERC.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python — Analyzing Text with the Natural Language Toolkit*. O’Reilly Media.
- H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva. 2013. Getting more out of biomedical documents with GATE’s full lifecycle open source text analytics. *PLoS Comput Biol*, 9(2).
- Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. [Table filling multi-task recurrent neural network for joint entity and relation extraction](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Y. Hong, Y. Liu, S. Yang, K. Zhang, A. Wen, and J. Hu. 2020. [Improving graph convolutional networks based on relation-aware attention for end-to-end relation extraction](#). *IEEE Access*, 8:51315–51323.
- Thomas N. Kipf and Max Welling. 2017. [Semi-supervised classification with graph convolutional networks](#). In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *52nd Annual Meeting of the Association for Computational Linguistics*.
- Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.
- Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.
- Changzhi Sun, Yeyun Gong, Yuanbin Wu, Ming Gong, Daxin Jiang, Man Lan, Shiliang Sun, and Nan Duan. 2019. [Joint type inference on entities and relations via graph convolutional networks](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1361–1370, Florence, Italy. Association for Computational Linguistics.
- Benjamin Therien, Parsa Bagherzadeh, and Sabine Bergler. 2021. CLaC-bp at MeasEval 2021. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*.
- Erik F. Tjong Kim Sang and Fien De Meulder. 2003. [Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition](#). In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.
- Liang Yao, Chengsheng Mao, and Yuan Luo. 2018. [Graph convolutional networks for text classification](#). *CoRR*, abs/1809.05679.
- Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *arXiv preprint arXiv:1802.09691*.
- Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *AAAI*, pages 4438–4445.

# CLaC-BP at SemEval-2021 Task 8: SciBERT Plus Rules for MeasEval

Benjamin Thérien, Parsa Bagherzadeh, and Sabine Bergler

CLaC Labs, Concordia University

Montreal, Canada

{b\_therie, p\_bagher, bergler} @cse.concordia.ca

## Abstract

This paper explains the design of a heterogeneous system that ranked eighth in competition in SemEval2021 Task 8. We analyze ablation experiments and demonstrate how the system components, namely tokenizer, unit identifier, modifier classifier, and language model, affect the overall score. We compare our results to similar experiments from the literature and introduce a grouping algorithm developed in the post-evaluation phase that increased our system’s overall score, hypothetically elevating our competition rank from eight to six.

## 1 Introduction

The MeasEval 2021 shared task involves identifying related groups of the four annotation types seen in Table 1, identifying relations linking the annotations of each group, and providing additional information (units and modifiers) for quantities (see (Harper et al., 2021)). Our system learns to classify tokens, groups tokens of the same class into spans and further groups related spans using a distance-based heuristic, providing a baseline for systems that attempt to learn these groupings.

## 2 Data

The data comprises excerpts from scientific articles of different disciplines: Astronomy, Engineering, Medicine, Agriculture, Biology, Chemistry, Earth Science, Computer Science, and Mathematics. The limited data (see Table 1), the wide range of topics involved, and the idiosyncrasies of each scientific field’s vocabulary make the task difficult. In fact, the task organizers report a high degree of inter-annotator disagreement for certain annotation types. Krippendorff’s alpha (Hayes and Krippendorff, 2007) for each annotation type is also shown in Table 1, underlining that annotating Measured Properties, Measured Entities, and Qualifiers poses a substantial degree of ambiguity, even to humans.

Annotation	Frequency	$\alpha$
Quantity (QA)	1164	.94
Unit (U)		.86
MeasuredEntity (ME)	1148	.54
MeasuredProperty (MP)	742	.64
Qualifier (QL)	276	.33

Table 1: Frequency of training data annotations and their Krippendorff’s Alpha

The training data contains a total of 298 excerpts containing 1164 different quantities. We refer to groupings of annotations as complete data-points (see Figure 1).

All complete data-points contain one quantity and at most one annotation span of each other type. A summary of complete data-point frequencies is provided in Table 2.

Annotations present	Frequency
QA	11
ME,QA	361
ME,MP,QA	512
ME,QL,QA	50
ME,MP,QL,QA	225
MP,QA	4
MP,QL,QA	1

Table 2: Frequency of different complete data-points in the training data

## 3 System

Our system is a pipeline of different machine learning and rule-based modules. We use SpaCy<sup>1</sup> for sentence splitting and tokenization and we fine-tune a SciBERT model<sup>2</sup> to classify each token into one of *Quantity* (QA), *Measured Entity* (ME), *Measured Property* (MP), *Qualifier* (QL), or *Other* (O).

<sup>1</sup><https://spacy.io/>

<sup>2</sup>SciBERT uses the same model architecture and pretraining objective as BERT (Devlin et al., 2019), but it is pre-trained on a large corpus of scientific text (Beltagy et al., 2019).

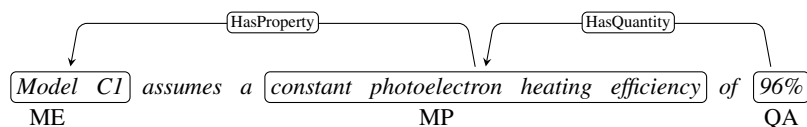


Figure 1: An example of a complete data point with corresponding annotations

During a first postprocessing phase, the token level output of the classifier is merged into spans of adjacent annotations of the same type. We feed the spans annotated as QA to a second SciBERT classifier to determine their modifier class (see Section 3.4). A distance-based heuristic groups the classified spans into complete data-points.

All of the system’s deep learning components are implemented using PyTorch (Paszke et al., 2017). Our models use simple (linear + softmax) classification on top of BERT’s implementation from the HuggingFace library.<sup>3</sup> Pre-trained weights are obtained from HuggingFace’s Model Hub.<sup>4</sup> Our competition SciBERT models are optimized by Adam (Kingma and Ba, 2015) using a constant learning rate of  $5e-6$  (the rest of Adam’s parameters are set to PyTorch’s defaults). The competition system’s token classifier is fine-tuned on all the training data for 5 epochs, while the modifier classifier is fine-tuned on all the training data for 3 epochs.

### 3.1 Preprocessing

**Tokenization** We modify SpaCy’s tokenizer to make several symbols individual tokens. As illustrated in Tables 3 and 4, our tokenizer separates apart mathematical symbols as well as suffixes *Rp*, *Rs*, *mH*, *RRh*.

Prefix	= $\sim \pm - \leq > < \geq$
Infix	= $\sim \approx \bullet : \% ( ) \rightarrow + - \pm , > <$
Suffix	<i>Rp</i> <i>Rs</i> <i>mH</i> <i>RRh</i>

Table 3: Tokenization rules added

The tokenization rules split apart tokens that contain annotations of different types (see Table 4).

**Gold span to token conversion** The token-level objective used to fine-tune SciBERT requires mapping the gold span annotations onto tokens. However, the gold standard annotations don’t always coincide with our token spans. Some gold annotation boundaries are in error (e.g. *oss rate* instead

<sup>3</sup><https://huggingface.co/transformers>

<sup>4</sup>The SciBERT models we refer to are fine-tuned from ‘scibert\_scivocab\_cased’ and the BERT model we refer to is fine-tuned from ‘bert-base-cased’.

SpaCy tokens	Modified tokens
$\sim$ 2-3, mA, /, cm2	$\sim$ , 2, -, 3, mA, /, cm2
$\sim$ 0.40-0.45, V	$\sim$ , 0.40, -, 0.45, V
$\sim$ 27%	$\sim$ -, 27, %
(ratio, CaP=1.67)	(,ratio, CaP, =, 1.67,)

Table 4: Tokenization examples

of *loss rate*) and some are deliberate (e.g. *beach* as the annotated part of the token *beaches*).

Of the 3330 gold spans that we convert to token-level training samples, only 48 are off by one character, and 2 are off by more than one character. The remaining token level annotations perfectly match the gold spans in character offset. We therefore convert gold annotations to token annotations naively, projecting gold span annotation onto the token span with the least character differences and obtain mappings as illustrated in Figure 2a.

**Sentence splitting** Scientific text makes frequent use of acronyms (e.g. *fig.*) which confuse SpaCy’s default sentence splitter. We use exclusion rules for frequent abbreviations and special tokens that end in punctuation. We also exclude symbols from starting a sentence, as well as tokens not preceded by a period.

### 3.2 Token Classification

We fine-tune SciBERT to classify tokens into the five output categories QA, QL, MP, ME, and O, where O indicates no annotation for a token. This module processes one document at a time, taking a list of sentences as input, where each sentence is a list of tokens. The output of the classifier is the token annotations for each of the sentences it receives.

**Input** The special token *[CLS]* is followed by tokenized sentences separated by *[SEP]*.

**Token classification** We use a linear layer followed by softmax for the token level multi-class classification into the five label categories from SciBERT output

**Out of vocabulary tokens** SciBERT’s Word-Piece tokenizer (Wu et al., 2016) splits unknown

tokens into *subwords* for which it has embeddings. We distribute a token’s gold annotation over all subwords within its span (see Figure 2b). To calculate the loss between gold standard and predicted tags, we use cross-entropy. Finally, to predict the class of a token that SciBERT breaks, we take the majority class of its subwords, breaking ties using the class of the first subword, contrary to (Devlin et al., 2019), who always use the class of the first subword.

### 3.3 Token Span Creation

After token classification, we group any adjacent tokens labelled with the same class into a single span for that label. If the label for a token differs from the label of its neighbors, it forms a span by itself.

### 3.4 Modifiers

In addition to the five classes discussed so far, MeasEval 2021 also annotates 10 modifier categories<sup>5</sup> for quantity spans: *IsApproximate*, *IsCount*, *IsRange*, *IsList*, *IsMean*, *IsMedian*, *IsMeanHasSD*, *IsMeanHasTolerance*, *IsRangeHasTolerance*, *HasTolerance*, or *NOMOD*.

For instance, the underlined quantity in Example 1 has a modifier class of *HasTolerance* triggered by  $\pm$ .

(1) ... constrain the CTB at 93.90 $\pm$ 0.15 *Ma*

To determine a quantity’s modifier class, we fine-tune a second SciBERT model. Using the *[CLS]* token as a representation for a quantity’s span, the model classifies the span into one of the 11 categories.

Our model predicts at most one modifier per quantity span, which fails in certain examples with more than one modifier. The training data contains 552 quantities with modifiers and 37 of those quantities have two modifiers assigned to them. In competition, our system ranked third for the modifier class.

### 3.5 Unit identification

We identify units in quantities using a simple rule-based algorithm. No unit is predicted when a quantity span ends in a number or when its predicted modifier is *IsCount*. Otherwise, mark the unit as the string of characters starting from the last occurring numerical character in the quantity span to the

<sup>5</sup>We add an 11th *NOMOD* which indicates the absence of a class.

end of that quantity span. Example 1 highlights the unit in a gray box.

## 3.6 Span Grouping

We present two approaches for grouping annotation spans into datapoints. The first approach is our original competition algorithm, the second is an improved, post-competition version. Each approach works from lists of token spans (as described above) and outputs a list of groupings. Groupings approximate complete data points.

### 3.6.1 Original

Our competition system creates candidate groupings for each quantity in a first pass by adding at most one measured entity, one measured property, and one qualifier span to the group if they occur within the same sentence as the group’s quantity. A span cannot be assigned to multiple groups. Next, we calculate the character distance between each group’s quantity and any still unmatched annotations that are at most one sentence away. The list of distances is sorted and the closest missing annotation within one sentence distance is added. While this algorithm creates some correct complete datapoints, it has two major drawbacks: one measured entity cannot be used for multiple groups and we do not rank the matches to achieve the best fit for all the quantities. When used in our competition system, this algorithm generates 1626 True positives, 892 false positives, and 1504 false negatives.

### 3.6.2 Span++

In the post evaluation phase, we tested a new algorithm that accounts for token distance during the grouping process. First, the algorithm initializes candidate groups for each quantity. Then, each quantity span is paired with each measured entity span and the shortest token distance between the spans is calculated. The measured entity from the closest pair is added to its quantity’s group and the pair is removed from the list. This is repeated for all measured entities that are within a 68 token distance from their quantity. Measured entities farther away are discarded. Then, the same matching process is used for measured properties and qualifiers with cutoff distances of 26 and 25 tokens respectively. Using identical settings to our original submission, we evaluate the performance of our new algorithm (called ‘span++’ in Table 5). We obtain consistently better overlap f1-scores. While this algorithm supersedes its predecessor, it still



a)	Tokens:	<i>Model</i>	<i>Cl</i>	<i>assumes</i>	<i>a</i>	<i>constant</i>	<i>photoelectron</i>	<i>heating</i>	<i>efficiency</i>	<i>of</i>	<i>96</i>	<i>%</i>	
	Gold tags:	ME	ME	O	O	MP	MP	MP	MP	O	QA	QA	
b)	SciBERT tokens:	<i>Model</i>	<i>Cl</i>	<i>assumes</i>	<i>a</i>	<i>constant</i>	<i>photo</i>	<i>##electron</i>	<i>heating</i>	<i>efficiency</i>	<i>of</i>	<i>96</i>	<i>%</i>
	Expanded tags:	ME	ME	O	O	MP	MP	MP	MP	MP	O	QA	QA

Figure 2: (a) Converting span annotations of Figure 1 to token annotations. (b) If SciBERT tokenizer breaks a token, we assign its gold tag to all of its sub-tokens

fails to account for multiple quantities having the same measured entity. When used in our competition system, this algorithm generates 1728 True positives, 778 false positives, and 1402 false negatives.

### 3.7 Relations

Our system uses predefined rules to determine the relations between annotations in a data-point. If there is a measured entity and no measured property, then the measured entity HasQuantity, otherwise, when both are present, the measured entity is assigned HasProperty and the measured property is assigned HasQuantity. Anytime a qualifier is added to a span, we stipulate that it qualifies the quantity.

## 4 Results

Our submission ranked eighth in competition (listed as rank 7 on the CodaLab leaderboard). Our system is robustly above average on all categories (see Table 5) with strong performance in quantity overlap score (tied for 2nd), qualifier overlap score (2nd), modifier overlap score (3rd), and qualifies overlap score (3rd).

Table 5 shows competition and post-competition results. Our competition system is called *CLaC-BP* and the competition winner is listed under *top ranked*. The first six entries of Table 5 show results for ablation experiments.

Our competition system had a non-zero dropout probability, therefore, we cannot recreate its exact performance for the benchmarking of our ablations. The system labelled **Full** was run with zero dropout probability and is otherwise identical to our competition system. All ablated systems are compared to this baseline.

**Noticeable differences** The first comparison in row 2 substitutes our fine-tuned SciBERT model for a fine-tuned BERT base model for token classification. SciBERT outperforms BERT for every

category except quantity and unit.

In row 3 we assess our modified tokenizer, by substituting it with SpaCy’s default tokenizer. The performance is near identical to our baseline system in every category.

The third and fourth systems were trained without modifiers and without units respectively. Overall, the system without units performed worse than the system without modifiers. This is, however, to be expected as there are 905 units and only 552 modifiers, i.e. units account for more overlap score.

Finally, the last comparison showcases the improvements when using *span++*.

## 5 Discussion

Our system’s modular pipeline allows us to assess the components in ablation studies and the various experiments we perform are informative about the task.

Compared to BERT base, SciBERT pretraining gives a solid advantage to all categories, except quantities and units. Not surprisingly, Table 5 shows a greater increase in precision than recall and the exact matches (EM) and F1 overlap (O) scores rise significantly.

Comparing SciBERT’s and BERT’s performance on the token level objective (Table 6) used during fine-tuning, we see that SciBERT yields greater precision for all competition categories, while BERT yields higher recall for all but *other*. In addition, the token level evaluation is not fully commensurate with the task evaluation, as the significant second task of grouping different spans is not fully assessed.

Our system might benefit from a more precise token level classifier due to its grouping algorithm: it is reasonable to assume that it performs better when more of the detected spans are correct (high precision system) and performs worse with a greater number of incorrect spans (high recall system).

Substituting SpaCy’s tokenizer into our system does not make much difference because SciBERT’s WordPiece tokenizer will break any unknown to-

System	LM	EM	P	R	F	O	QA	ME	MP	QL	Unit	Mod	HQA	HP	Quals
Full	SciBERT	.346	.641	.516	.572	.382	.848	.249	.308	.111	.667	.549	.296	.136	.061
Full	BERT <sub>base</sub>	.311	.591	.496	.539	.349	<b>.853</b>	.239	.24	.063	<b>.697</b>	.537	.253	.107	.038
Full no M.T*	SciBERT	.341	.644	.513	.571	.381	.848	.258	.306	.103	.653	.522	.297	.148	.056
Full no Mod	SciBERT	.309	.63	.464	.534	.347	.848	.249	.308	.111	.667	.0	.296	.136	.061
Full no Unit	SciBERT	.277	.608	.422	.498	.314	.848	.249	.308	.111	.0	<b>.567</b>	.296	.136	.061
Full with span++	SciBERT	<b>.38</b>	<b>.687</b>	<b>.551</b>	<b>.612</b>	<b>.421</b>	.848	<b>.292</b>	<b>.367</b>	<b>.151</b>	.667	.549	<b>.371</b>	<b>.162</b>	<b>.081</b>
CLaC-BP (rank 7)	SciBERT	–	–	–	–	.389	.855	.251	.318	.107	.677	.546	.308	.147	.058
top ranked	–	–	–	–	–	.519	.861	.437	.467	.163	.722	.642	.482	.318	.092
Compet. mean	–	–	–	–	–	.323	.741	.200	.196	.021	.602	.364	.205	.114	.012
Compet. median	–	–	–	–	–	.369	.818	.251	.245	.000	.661	.375	.308	.147	.000

\*M.T: Modified tokenizer EM: Exact match HQA: HasQuantity HP: HasProperty Quals: Qualifies

Table 5: Ablation and competition results

kens, however they are tokenized, into subwords at inference time. Yet, because the classifier output is projected onto input tokens, SpaCy’s failure to separate certain tokens (see Table 4) might be responsible for the very small degradation when using the SpaCy tokenizer.

Addressing the modifier categories with a separate SciBERT model is successful and allows our system to benefit from several rule-based transition phases. While the task is not well understood (see the overall low performances and the high inter-annotator disagreement), we believe this type of architecture to be more beneficial than end-to-end systems, as we can pinpoint weaknesses and experiment more easily. This is demonstrated by the sizeable improvement of results when implementing a simple distance measure constraint for the grouping step. Once such features have been identified as effective, their encoding in the classifier becomes possible.

Class	P		R		F1		support
	S	B	S	B	S	B	
ME	<b>.43</b>	.33	.49	<b>.55</b>	<b>.46</b>	.41	457
MP	<b>.47</b>	.46	.55	<b>.61</b>	.51	<b>.53</b>	300
QA	<b>.84</b>	.81	.96	<b>.98</b>	<b>.90</b>	.88	950
QL	<b>.45</b>	.34	.23	<b>.32</b>	.30	<b>.33</b>	240
o	<b>.98</b>	.98	<b>.97</b>	.94	<b>.97</b>	.96	12885
<i>M<sub>avg</sub></i>	<b>.63</b>	.59	.64	<b>.68</b>	<b>.63</b>	.62	14832

Table 6: Validation SciBERT (S) and BERT base (B) validation set performance after 4 epochs of fine-tuning

## 6 Summary

Measurements are ubiquitous in scientific articles, yet have so far not been addressed. The MeasEval task is an opportunity to combine different subtasks and experiment how best to combine them. Our system approaches the MeasEval task in a modular fashion: preprocessing, two classifications, and postprocessing. It breaks the task into two, first a

classification of the relations for measured entities, measured properties, and qualifiers and second a classification of modifiers. SciBERT, trained on scientific articles, yields better performance than BERT in our experiments, mainly due to improvements in precision. The differences, however are small. Before, between, and after the two classification steps are a number of rule-based modules that create the classifier input and piece together the classifier output for submission. Several experiments on variations in these rule-based accessories suggest their usefulness and ways to improve our results.

## References

- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Andrew F. Hayes and Klaus Krippendorff. 2007. [Answering the call for a standard reliability measure for coding data](#). *Communication Methods and Measures*, 1(1):77–89.
- Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic optimization. In *Proceed-*

*ings of the 3rd International Conference on Learning Representations, ICLR'15.*

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS 2017*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google's neural machine translation system: Bridging the gap between human and machine translation](#). *CoRR*, abs/1609.08144.

# THiFly\_Queens at SemEval-2021 Task 9: Two-stage Statement Verification with Adaptive Ensembling and Slot-based Operation

Yuxuan Zhou<sup>1</sup>, Kaiyin Zhou<sup>2,3</sup>, Xien Liu<sup>1</sup>, Ji Wu<sup>1</sup>, Xiaodan Zhu<sup>4</sup>

<sup>1</sup>Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

<sup>2</sup>THiFLY Research, Tsinghua University, Beijing 100084, China

<sup>3</sup>State Key Laboratory of Cognitive Intelligence, Hefei, Anhui 230088, China

<sup>4</sup>ECE & Ingenuity Labs Research Institute, Queen's University

zhouyx17@mails.tsinghua.edu.cn kyzhou3@iflytek.com

{xeliu, wuji\_ee}@mail.tsinghua.edu.cn

xiaodan.zhu@queensu.ca

## Abstract

This paper describes our system for verifying statements with tables at SemEval-2021 Task 9. We developed a two-stage verifying system based on the latest table-based pre-trained model GraPPa. Multiple networks are devised to verify different types of statements in the competition dataset and an adaptive model ensembling technique is applied to ensemble models in both stages. A statement-slot-based symbolic operation module is also used in our system to further improve the performance and stability of the system. Our model achieves second place in the 3-way classification and fourth place in the 2-way classification evaluation. Several ablation experiments show the effectiveness of different modules proposed in this paper.

## 1 Introduction

Verifying whether a statement is entailed, refuted, or unknown with the given table is a challenging task, which requires the system to understand the statement and table jointly and reasons from the two information resources. The studies on this task could benefit several downstream applications, e.g. fake news detection. Recently, with the release of a large-scale dataset for table-based fact verification named TABFACT (Chen et al., 2019), this task received several studies. Verifying statements based on tables is a challenging task since the researches on understanding tables are not enough compared with works on free-texts, and the methods to train models understanding free-text and table jointly need further studies as well.

While TABFACT contains a huge number of statements and tables, the tables in the TABFACT dataset are relatively simple since they do not have hierarchical column heads as tables in scientific papers do and the contents in the tables are easier to understand compared to the tables in scientific

papers as well. In the SemEval task 9 (Wang et al., 2021), the goal is to develop a system that can verify statements (**subtask A**) and find evidence (**subtask B**) based on the tables extracted from scientific tables. Our team is more interested in the verifying task and only participated in subtask A. Different from data in TABFACT, in subtask A, we are also required to classify a new type of statement that cannot be entailed or refuted based on the given table, named “unknown” type. Since no statements of this type are given in the training data, the classification of this type of statement becomes a core difficulty of the subtask.

This paper describes our two-stage table-based verifying system based on the latest table-based pre-trained language model GraPPa (Yu et al., 2020). The system leverages the model ensembling technique to ensemble different verifying models which are designed to solve different types of statements in the dataset in both two stages of the system. The statement-slot technique is also used to capture and solve a small part of the data by symbol calculation, which helps to increase the performance and stability of our system. Our system achieves a two-way score of 84.55 and a three-way score of 83.76 in subtask A respectively.

## 2 Related Work

### 2.1 GraPPa

GraPPa is a pre-trained model for table-based semantic parsing task, proposed by Yu et al. (2020). It is pre-trained on the synthetic question-SQL pairs with a novel text-schema linking objective, where the SQL queries are generated by a synchronous context-free grammar(SCFG). The text-schema linking objective makes the model predict the syntactic roles of the columns in the SQL to encourage the model to notice the link between

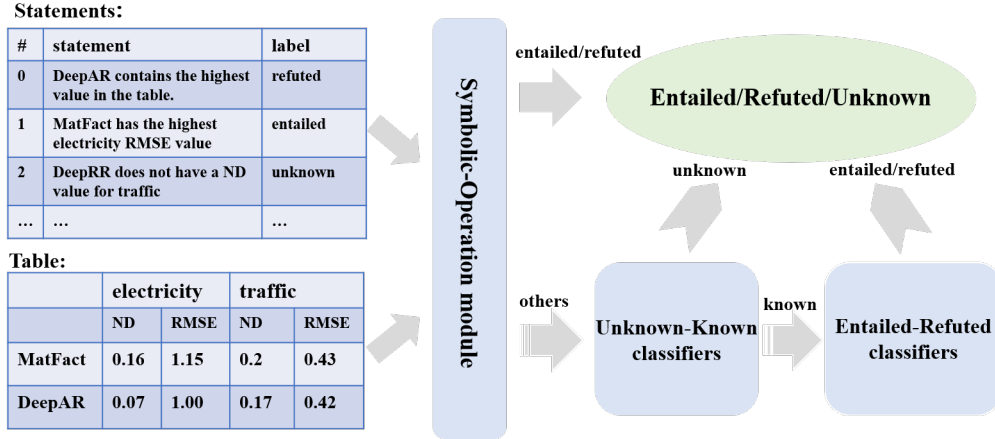


Figure 1: The flowchart of the system. Statements and tables are first sent into the symbolic-calculation module, where statements with simple sentence structures are verified by symbol calculation. Statements that are not processed by the calculator are then sent into the two-stage deep-learning-based verifying system, where the first stage’s models verify whether the statement cannot be verified by the given table, and the second stage’s models verify whether the statement can be entailed by the given table.

natural language phrases and the corresponding logical form constituents.

The authors of GraPPa use 475k synthetic examples to pre-train GraPPa. On four popular semantic parsing benchmarks, GraPPa consistently achieves state-of-the-art results, which shows the powerful table understanding ability of GraPPa. In this work, we find that GraPPa’s ability of understanding tables can also be migrated to the table-based fact verification task and achieves great performance on this task.

## 2.2 Mixture-of-Experts layer

In Shazeer et al. (2017) the authors proposed a sparsely mixture-of experts layer applied on the stacked LSTM model which achieves new state-of-the-art results on language modeling and machine translation benchmarks with lower computational cost and larger model capacity. In this work, we applied this MoE layer on the top of the GraPPa model as a part of our verifying system.

## 3 System Description

This section mainly describes the details of the two-stage table-based verifying system. The first stage classifies unknown type statements from the other two types of statements, while the second stage further classifies the entailed type statements from refuted type statements. We find that the proposed two-stage system works better than a direct three-way classification system, because 1) no unknown type statements are provided in the

train set, which brings difficulty to the training process of the three-way classification system; 2) the two-stage system is closer to the processing procedure of human since you have to decide whether the table knowledge is enough to entail or refute the given statement before the further reasoning. Before the two-stage deep-learning-based system, a symbolic-calculation module is added to capture and process some statements with simple sentence structures. The symbolic-calculation module has the features of low recall and high precision and is used to process some numerical type statements (the verifying process involves numerical operations) since we find that the deep-learning-based system is unstable when processing such type of statements. The flowchart of the whole system is displayed in Figure 1.

### 3.1 Statement-table joint encoder

In both stages of our system, multiple binary classifiers are ensembled together to verify statements. All of the binary classifiers used in the system are constructed based on the table-based pre-trained model GraPPa. In the system, the GraPPa model works in two way: 1) encoding the statement and the pruned table by following the table-BERT method and the table pruning algorithm proposed in Chen et al. (2019); 2) encoding the statement and each row of the table separately by the same method. While the first way is the standard encoding method of NLI tasks, we found the second way

also has its advantages and will explain later. After encoding the table and statement by GraPPa, multiple networks are devised to do reasoning based on the encoded representations. The following sections provide further details of these models, named pruned-table-based models and whole-table-based models separately.

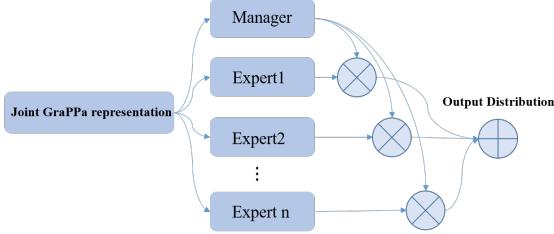


Figure 2: The structure of verifying model is based on the types of statements. An MLP manager activated by the softmax function takes the joint representation as input and outputs the probability distribution that the statement belongs to different types of statements. Multiple MLP experts process the joint representation and output the classification logits, which are further averaged by the statement type probability from the manager to achieve the final classification logits.

### 3.2 Pruned-table-based models

As introduced before, we simply substitute BERT by GraPPa in the Table-BERT method to get the joint representation of the statement and table. Multiple networks are devised to process the joint representation: 1) a simple linear layer, same as Table-BERT; 2) three MLP processing different types of statements (statements that need to count rows/columns, statements related with the superlative operation, other statements) with a softmax activated MLP manager outputs the probability that the statement belongs to each type of statements. The final output distribution is the weighted average of different MLP experts’ output distribution; 3) same structure as 2), but the statements are separated based on the language style (natural/artificial) and the MLP manager is trained based on the natural statement ids provided in the train set; 4) sparsely MoE layer introduced in Shazeer et al. (2017) with applying the implementation on <https://github.com/davidmrau/mixture-of-experts>. Figure 2 shows the basic model structure of networks 2) and 3).

### 3.3 Whole-table-based models

As serialized tables obtained by language template are usually too long to be encoded by transformer-

based model, the above methods use a table pruning algorithm to choose statement-related columns from the table, while the algorithm is not perfect and it may discard some useful columns in tables. Even if the pruning algorithm chooses all the related columns, the pruned serialized table may still be too long. The whole-table-based models regard the table as the set of rows and serialize each row by the same method. Then each row is concatenated with the statement and sent in GraPPa to get the representation of tokens in the statement and each row. Two different reasoning networks are applied after the encoding process. The first one is a simple attention network across all tokens in the statement and table, the final output distribution is produced by the following:

$$Q = RR^T, Q' = \text{softmax}(Q), F = Q'R \quad (1)$$

$$\mathbf{att} = RW_a + \mathbf{b} \quad (2)$$

$$\mathbf{out} = \text{MLP}(F^T \mathbf{att}) \quad (3)$$

where  $R \in \mathbb{R}^{n \times e}$  is the representation matrix of all the tokens in the table and statements,  $n$  is the product of row numbers and the maximum token numbers of the concatenation between the statement and different rows,  $e$  is the embedding size. Softmax is applied to the attention matrix, where  $Q'_{ij}$  represents the  $j^{\text{th}}$  token’s contribution to the  $i^{\text{th}}$  token.  $F$  is the attended representation matrix, and  $\mathbf{att}$  is the importance score of tokens in statement and table achieved by a linear layer where  $W_a \in \mathbb{R}^{e \times 1}$  and  $\mathbf{b} \in \mathbb{R}^{e \times 1}$  are trainable parameters. Follow the equation 3, the representations are further aggregated by the importance score and sent into an MLP classifier to get the output distribution.

Another choice of reasoning network is the GAT network proposed in Liu et al. (2019) to aggregate evidence from different sources. GAT hierarchically aggregates information (first in token-level and then in sentence-level) which is more reasonable compared to the simple attention network mentioned before. For the convenience of presentation, all the models mentioned in this section are named and listed in Table 1.

### 3.4 Adaptive model ensembling

An adaptive model ensembling technique is applied to both stages of our verifying system, where

Model	Description
Table-GraPPa	GraPPa with a linear layer
NumGuide	GraPPa with three MLP experts for verifying different numerical type statements
StmtGuide	GraPPa with two MLP experts for verifying statements with different language styles
MoE	GraPPa with sparsely MoE layer (8 MLP experts)
RowAtt	GraPPa with attention layer across all tokens
RowGAT	GraPPa with GAT aggregation module

Table 1: Name and description of models used in the system

different weights of models are searched in different subsets of the development set. Such ensembling technique works well in our system since models work differently on different types of data (e.g. A model may perform better on shorter statements while B model may perform better on longer statements). In the first stage, the weight of each model is searched on two subsets of development set based on the length of the statement and serialized table (we found that some models have a better performance on longer input sequences than others, and set 300 words as the threshold).

In the second stage, the type of each statement is first recognized by some language features and then the model weights are searched on each subset of the development set. Three types of statements are defined: 1) “count” type statements, in which the model needs to count the specific rows or columns in the table; 2) “superlative” type statements, in which the model need to find the maximum or the minimum value of one row/column; 3) “same/different” type statements, which the model need to decide whether some cells’ value are the same or different. The first two types of statements are recognized by statement slots, while the third type is recognized by trigger words (“same”, “different”, “equal”). We apply the weights searched from the development set on these three statement types and simply do averaging ensemble to the rest of the statements. The ensembling weights are searched on the development set without the participation of models trained with train and development set, while we replaced some models with the version that retrained on both train and development set in the evaluation period and direct applied the weights searched on the development set. More details of the models involved in ensembling are presented in the appendix.

Type	Statement slot
Count	there be () value(s) in the table
Count	there be () different ()
Superlative	() lowest () in the table be ()
Superlative	() has highest value(s) of ()

Table 2: Examples of the statement slots used in the system

### 3.5 Statement-slot based symbolic-calculation module

To further increase the performance and the stability of the system, a statement-slot-based symbolic-calculation module is developed to solve some numerical type statements which are relatively difficult to deep models. The symbolic-calculation module is added before the two-stage system with a low recall and high precision. Several statement slots are devised to capture “count” and “superlative” type statements. After the capture, the symbolic-calculation module parses the table and extracts the corresponding rows/columns based on a designed entity linking algorithm, and then do the related logical calculation based on the category of the statement and the parsing result of the table. Table 2 shows some examples of the statement slots used in our system.

### 3.6 Training method

In the first stage, the unknown type training data are created from three source: 1) from other tables’ statements under the same XML file, 1685 statements in total; 2) automatic generated statements by language templates, 1378 statements in total; 3) the statements from the related scientific papers (extracted from the related papers of training set tables, near the references of the tables by programming), 1272 statements in total. All of the statements in the train set are used as the known type data to train

models. Training processes are stopped when models reach the highest accuracy on the development set.

In the second stage, we first use the train set to train models and stop the training process at the maximum development set accuracy checkpoint and the minimum development set loss checkpoint. Then we add the development set into the train set and retrain some of the models, stopping at a fixed epoch to make the most use of data.

Almost all of the models are trained with the cross-entropy loss function except the NumGuide, StmtGuide, and MoE. For the NumGuide, we use trigger words to recognize the “counting” and “superlative” types statements in the train set and use the recognition result as labels to calculate the cross-entropy between the manager’s output and labels. For StmtGuide, we used the provided natural statement ids in the train set to calculate the cross-entropy between the manager’s output and labels. For MoE, an extra loss mentioned in Shazeer et al. (2017) is applied to avoid the local optimum. The extra loss functions mentioned above are simply added on the origin cross-entropy loss with a weight of 0.01. Models in the second stage are pre-trained on the TABFACT dataset before further trained on the competition dataset. More details about model training can be found in the appendix.

## 4 Evaluation

To evaluate our proposed verifying system, we perform two ablation studies regarding the adaptive ensembling method and the statement-slot-based symbolic-calculation module used in our system on both the development set and test set.

Table 3 shows the evaluation of our system on both development set and test set, compared with different ways of ensembling. We found that the adaptive ensembling on both stages achieves the highest two-way and three-way scores on the development set, while it only improves the three-way classification performance on the test set and slightly regresses on the two-way classification. We assume that it may be because of the difference between the data distributions of the development set and the test set.

Besides the ablation experiment on ensembling, we also perform an ablation experiment on the statement-slot-based symbolic-calculation module. Table 4 shows the result of the ablation experiment. The result of the experiment shows that the pro-

posed symbolic-calculation module increases the performance on both the development set and test set by around 5 percent, which shows the benefit of the symbolic-calculation module on verifying numerical type statements.

Ensembling type	Dev set		Test set	
	2-way	3-way	2-way	3-way
W+W	<b>87.81</b>	<b>86.93</b>	84.55	<b>83.76</b>
W+A	86.77	86.09	84.92	82.45
A+W	86.50	84.31	<b>85.41</b>	81.58
A+A	85.46	83.47	85.22	81.41

Table 3: Ablation experiment regarding the ensembling in the two stages of the system, where **W** refers to the adaptive ensembling and **A** refers to the simple averaging ensembling. The sequence of the letter refers to the ensembling setting of two stages, e.g. W+A means the first stage applies adaptive ensembling and the second stage applies average ensembling.

Ensembling type	Dev set		Test set	
	2-way	3-way	2-way	3-way
w/ sym-cal	<b>87.81</b>	<b>86.93</b>	<b>84.55</b>	<b>83.76</b>
w/o sym-cal	82.31	82.36	78.94	77.79

Table 4: Ablation experiment regarding the symbolic-calculation module (sym-cal). The result of the experiment shows that the proposed symbolic-calculation module consistently improves the performance on both the development set and the test set.

## 5 Conclusion

This paper describes our two-stage verifying system developed for SemEval-2021 Task 9, which leverages the latest table-based pre-trained model GraPPa. The two-stage verifying structure allows us to develop more targeted models on both stages of the system. Multiple reasoning networks are applied behind the GraPPa model, and an adaptive model ensembling technique is used in both stages of the system. A statement-slot-based symbolic-calculation module is also added at the top of the whole system to further improve the performance and stability of the system. Ablation experiments show the effectiveness of the methods proposed in the paper.

## References

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and



- William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.
- Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2019. Fine-grained fact verification with kernel graph attention network. *arXiv preprint arXiv:1910.09796*.
- Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 Task 9: A fact verification and evidence finding dataset for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of SemEval*.
- Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *arXiv preprint arXiv:2009.13845*.

## A Weight searching in adaptive ensembling

We use a grid search method to search the weight of each model in adaptive ensembling. The searching algorithm can be expressed as follows:

1. Initialize total weight  $T=15$  (larger  $T$  may cause overfitting on the dev set and larger searching cost as well).
2. Generate all possible weight combinations  $L$ , where every element  $\mathbf{W}$  in  $L$  needs to satisfy the constraint:  $\sum_{i=1}^n W_i = T$  where  $n$  is the number of models joining in the ensembling.
3. For each  $\mathbf{W}$  in  $L$ , apply the normalized weights  $\mathbf{W}/T$  on models' results on the dev set to get the accuracy of ensembling. Save the set of weights with the highest accuracy as the best weights. If multiple sets of weights achieve the same best accuracy, calculate the variance of each set of weights and choose the set with the lowest variance as the best weights.

## B Models involved in adaptive ensembling

The following sections introduce the details of models involved in the two-stage ensembling. We revised the entity linking algorithm proposed in [Chen et al. \(2019\)](#) to get better-pruned tables. The new algorithm works better on some models, while we found some models with the origin entity linking algorithm also perform well and we keep them as a part of ensembling. In the following description, the model applies the revised entity linking algorithm if not mentioned specially.

For the simplicity of presentation, we define the following usage of symbols: a "+" symbol added behind the name of the model means the model is trained on train set and stopped when reaching the maximum accuracy on dev set; a "-" symbol added behind the name of the model means the model is trained on train set and stopped when reaching the minimum loss on dev set; no symbol added behind the name of the model means the model is trained on both train and dev set and stopped at fixed epochs.

### B.1 Models involved in the first stage's ensembling

A total of 8 models participate in the first stage's ensembling, while some models are trained by bi-ased CE loss (we adjusted the weight of different classes with 1.5:1 for unknown and other classes) to reach a more balance recall and precision. They are: 1) Table-GraPPa+ ;2) RowGAT+; 3) RowAtt+; 4) MoE+; 5) StmiGuide+; 6) RowAtt+ trained with weight added to the loss function; 7) Table-GraPPa+ trained with weight added to the loss function; 8) RowGAT+ trained with weight added to the loss function.

### B.2 Models involved in the second stage's ensembling

A total of 9 models participate in the second stage's ensembling, they are: 1) Table-GraPPa; 2) StmtGuide; 3) Table-GraPPa+ with origin entity linking algorithm; 4) MoE; 5) RowGAT; 6) NumGuide; 7) RowAtt; 8) MoE-; 9) Table-GraPPa with origin entity linking algorithm.

When searching the ensembling weights on the dev set, we simply do the following replacements: change Table-GraPPa with Table-GraPPa+; change StmtGuide with StmtGuide+; change MoE with MoE+; change RowGAT with RowGAT+; change NumGuide with NumGuide+; change RowAtt with RowAtt+; change Table-GraPPa with origin entity linking algorithm with Table-GraPPa- with origin entity linking algorithm. Ensembling weights are searched on these models without training on the dev set and directly apply to the corresponding model trained on both train and dev sets.

## C More details about model training

All models are trained with AdamW optimizer (implemented by Huggingface) with a warmup ratio be 0.3 and learning rate be  $2e-5$ , and all models in the second stage are pre-trained first on the TABFACT dataset.

# TAPAS at SemEval-2021 Task 9: Reasoning over tables with intermediate pre-training

Thomas Müller, Julian Martin Eisenschlos, Syrine Krichene

Google Research, Zürich

{thomasmueller, eisenjulian, syrinekrichene}@google.com

## Abstract

We present the TAPAS contribution to the Shared Task on Statement Verification and Evidence Finding with Tables (SemEval 2021 Task 9, Wang et al. (2021)). SEMTABFACT Task A is a classification task of recognising if a statement is entailed, neutral or refuted by the content of a given table. We adopt the binary TAPAS model of Eisenschlos et al. (2020) to this task. We learn two binary classification models: A first model to predict if a statement is neutral or non-neutral and a second one to predict if it is entailed or refuted. As the shared task training set contains only entailed or refuted examples, we generate artificial neutral examples to train the first model. Both models are pre-trained using a MASKLM objective, intermediate counter-factual and synthetic data (Eisenschlos et al., 2020) and TABFACT (Chen et al., 2020), a large table entailment dataset. We find that the artificial neutral examples are somewhat effective at training the first model, achieving 68.03 test F1 versus the 60.47 of a majority baseline. For the second stage, we find that the pre-training on the intermediate data and TABFACT improves the results over MASKLM pre-training (68.03 vs 57.01).

## 1 Introduction

Recently, the task of Textual Entailment (TE) (Dagan et al., 2005) or Natural Language Inference (NLI) (Bowman et al., 2015) has been adapted to a setup where the premise is a table (Chen et al., 2020; Gupta et al., 2020). The Shared Task on Statement Verification and Evidence Finding with Tables (SemEval 2021 Task 9, Wang et al. (2021)) follows this line of work and provides a new dataset consisting of tables extracted from scientific articles and natural language statements written by crowd workers. In this paper, we discuss a system for tackling task A, which is a multi-class classification task that requires finding if a statement is

entailed, neutral or refuted by the contents of a table. The training set contains only entailed and refuted examples and requires data augmentation to learn the neutral class. Additionally, this data set is composed of English language data and requires sophisticated contextual and numerical reasoning such as handling comparisons and aggregations.

A successful line of research on table entailment (Chen et al., 2020; Eisenschlos et al., 2020; Gupta et al., 2020) has been driven by BERT-based models (Devlin et al., 2019). These approaches reason over tables without generating logical forms to directly predict the entailment decision. Such models

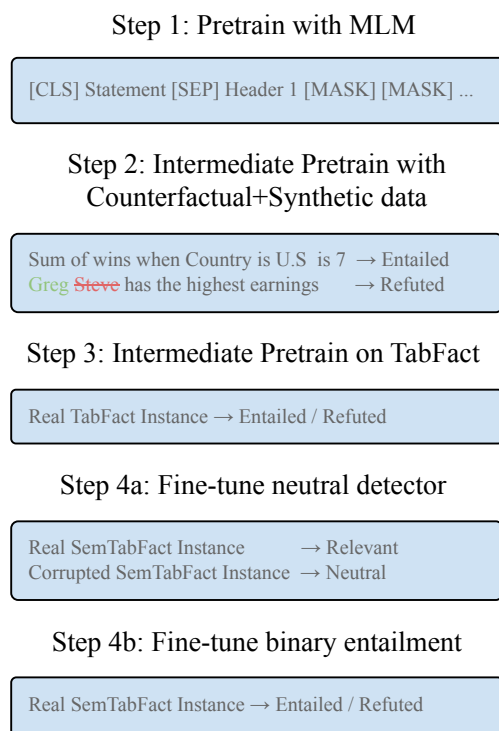


Figure 1: Overview of the training pipeline use in our system. We use intermediate pre-training on *Counterfactual+Synthetic* data (Eisenschlos et al., 2020) and then fine-tune on TABFACT (Chen et al., 2020).

are known to be efficient on representing textual data as well reasoning over semi-structured data such as tables. In particular, TAPAS-based models (Herzig et al., 2020) that encode the table structure using additional embeddings, have been successfully used to solve binary entailment tasks with tables (Eisenschlos et al., 2020).

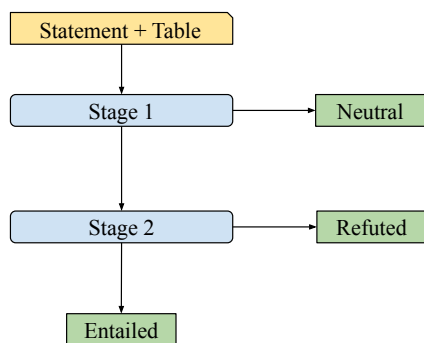


Figure 2: Overview of the complete system. Stage 1 classifies into neutral and non-neutral statements. Stage 2 into entailed and refuted. Both stages are based on binary TAPAS classifier models.

To address multi-class classification entailment we decompose the main task into two sub-tasks and use two TAPAS models as described in Figure 2. A first model classifies the statement into neutral or non-neutral, and a second into entailed or refuted. The two models are learned separately: we created artificial neutral statements to fine-tune the first model. Examples are extracted by randomly pairing statements and tables from the SEMTABFACT training set. We also generate harder examples by creating new tables from the original tables by removing columns that contain evidence to refute or entail the statement. This procedure is discussed in Section 3.

We follow Eisenschlos et al. (2020) and pre-train the two TAPAS models with a MASKLM objective (Devlin et al., 2019) and then with counterfactual and synthetic data as shown in Figure 1. We additionally fine-tune both models on the TABFACT dataset. Details are given in Section 4.

We find that our artificial neutral statement creations out-performs a majority baseline and that pre-training help for both the first and the second stage. Our best models achieve 68.03 average micro f1-score on the test set.

## 2 Related Work

**Entailment on Tables** Recognizing textual entailment (Dagan et al., 2010) has expanded from a text only task to incorporate more structured data, such knowledge graphs (Vlachos and Riedel, 2015), tables (Jo et al., 2019; Gupta et al., 2020) and images (Suhr et al., 2017, 2019).

The TABFACT dataset (Chen et al., 2020) for example, uses tables as the *premise*, or source of information to resolve whether a statement is entailed or refuted. The TAPAS architecture introduced by Herzig et al. (2020) can be used to obtain transformer-based baselines, as shown in Eisenschlos et al. (2020), by using special embeddings to encode the table structure. Zhang et al. (2020); Chen et al. (2020) also use BERT like models but obtain less accurate results due possibly to not using table-specific pre-training.

**Intermediate Pre-training** Our system relies on intermediate pre-training, a technique that appears in different forms in the literature. Language model fine-tuning (Howard and Ruder, 2018), or domain adaptive pre-training (Gururangan et al., 2020) are useful applications for domain adaptation. In a similar manner than Pruksachatkun et al. (2020), we use the *Counterfactual+Synthetic* tasks from Eisenschlos et al. (2020) to improve the discrete and numeric reasoning capabilities of the model for Table entailment.

**Synthetic data** The use of synthetic data to improve learning in NLP is ubiquitous (Alberti et al., 2019; Lewis et al., 2019; Wu et al., 2016; Leonardya et al., 2019). Salvatore et al. (2019) focus on textual entailment and probes models with synthetic examples. In semantic parsing Wang et al. (2015); Iyer et al. (2017); Weir et al. (2020) use templates to augment the training data for text-to-SQL tasks and Geva et al. (2020) do so to improve numerical reasoning, as do Eisenschlos et al. (2020) on tabular data. They also create minimal contrastive examples (Kaushik et al., 2020; Gardner et al., 2020) by automatically swapping entities in the statements by plausible alternatives that exists elsewhere in the table.

## 3 System

Our system is a two stage process that first decides whether a statement is neutral, and then decides if non-neutral statements are entailed or refuted. Both stages are implement using a binary TAPAS

Dataset	Statements	Tables	Entailed	Refuted	Neutral
Crowdsourced Train	4,506	981	2,818 (62.54%)	1,688 (37.46%)	
Auto-generated Train	179,345	1,980	92,136 (51.37%)	87,209 (48.63%)	
Stage 1 train	9,012	1,915	4506 (50%)		4506 (50%)
Dev	556	52	250 (44.96%)	213 (38.31%)	93 (16.73%)
Test	653	52	274 (41.96%)	248 (37.98%)	131 (20.06%)

Table 1: SEMTABFACT (Wang et al., 2021) statistics. The training data for the first stage was created from the crowdsourced training data using artificial neutral statements created by deleting columns with evidence or swapping statements randomly. For the second stage we use the crowdsourced training data.

classifier. TAPAS (Herzig et al., 2020; Eisenschlos et al., 2020) is a variation of BERT (Devlin et al., 2019), extended with special token embeddings that give the model a notion of the row and column a token is located in and what is its numeric rank with respect to the other cells in the same column.

### 3.1 Pre-training

The original TAPAS model (Herzig et al., 2020) was pre-trained with a Mask-LM objective (Devlin et al., 2019) on tables extracted from Wikipedia. It was later found (Eisenschlos et al., 2020) that its reasoning capabilities can be improved by further training on artificial counter-factual entailment data. This led to substantial improvements on the TABFACT dataset (Chen et al., 2020), a binary table entailment task similar to SEMTABFACT. On that dataset the test set accuracy for a BERT-based model improved from 69.6 to 78.6. In this work, we use models fine-tuned on TABFACT as the foundation for both stages. We also experimented with using models fine-tuned on INFOTABS (Gupta et al., 2020) and SQA (Iyyer et al., 2017) as the initial models but did not find that to achieve better accuracy. The overall pre-training strategy is described in Figure 1, where we also show how we use these checkpoints to use the two classification models described below.

### 3.2 Neutral Identification Stage

As discussed, the first stage of the system identifies if a statement is neutral. Training a system for this task is challenging as the SEMTABFACT training data does not contain neutral statements. We therefore created artificial neutral statements from two sources. Following the recommendation of the shared-task organizers, we created neutral statement by randomly pairing statements from the training set with new tables. Additionally, we created

neutral statements by identifying columns that contained evidence for deciding whether a statement is entailed and then randomly removing one of these columns. Our assumption is that it should not be possible to decide whether the statement is entailed when an evidence column has been removed. We do not remove the first column of a table since that often contains the name of the row entries. In order to detect the columns containing the evidence, we trained an ensemble of 5 TAPAS QA models on the automatically generated SEMTABFACT training set. Note that the auto-generated data is generated from templates and in contrast to the crowdsourced training data does have evidence cell annotations. The models are trained to predict the binary entailment decision as well as the evidence cells at the same time, and are initialized using a TAPAS model fine-tuned on SQA. The model is trained to predict the binary entailment decision as well as the evidence cells at the same time. Our models take as input  $[CLS]s_1\dots s_n[SEP]t_1\dots t_m$  where  $s_1, \dots, s_n$  represents the tokenized statement and  $t_1, \dots, t_m$  the tokenized table. For each token  $t$  of the table the model outputs a score for the token to be an evidence for the statement  $S$ ,  $s(t \in S) \in \mathbb{R}$ . Additionally, it outputs the scores of the entailment decision using the  $[CLS]$  tokens  $s([CLS]) \in \mathbb{R}$ .

We use the same hyper-parameters as SQA (as discussed in Herzig et al. (2020)). We then run these models over the crowdsourced training data and for all examples where the majority of the models correctly predicts the entailment label, we extract all columns for which a majority of the ensemble predicted at least one evidence cell. Evaluation on the SEMTABFACT development set showed that the precision of this column selection process is 0.87 (87% of the extracted columns contain a reference cell). For each column, we then create a new artificial neutral example by removing the

Stage 1	Stage 2	Dev				Test			
		f1 2-way		f1 3-way		f1 2-way		f1 3-way	
		Median	Ensemble	Median	Ensemble	Median	Ensemble	Median	Ensemble
Majority	Majority	51.44		42.80		52.41		42.15	
Majority	TABFACT	<b>78.33</b> $\pm 0.45$	<b>80.25</b>	66.40 $\pm 0.66$	68.29	<b>75.33</b> $\pm 0.79$	<b>75.21</b>	60.64 $\pm 0.65$	60.47
MASKLM	TABFACT	74.98 $\pm 0.39$	78.38	70.81 $\pm 0.66$	72.80	74.32 $\pm 0.84$	74.84	67.76 $\pm 0.50$	67.67
BERT	TABFACT	75.54 $\pm 0.75$	77.01	70.33 $\pm 0.59$	72.04	72.73 $\pm 0.86$	73.18	66.15 $\pm 0.38$	67.70
Inter	TABFACT	75.77 $\pm 0.50$	78.28	<b>71.21</b> $\pm 0.34$	72.79	72.94 $\pm 0.88$	74.01	<b>67.99</b> $\pm 0.78$	67.98
TABFACT (drop)	TABFACT	78.02 $\pm 0.45$	80.06	67.88 $\pm 0.87$	69.47	74.92 $\pm 0.76$	75.02	62.41 $\pm 0.51$	61.67
TABFACT (random)	TABFACT	75.97 $\pm 0.73$	78.50	69.62 $\pm 0.93$	71.81	74.77 $\pm 0.97$	74.67	66.64 $\pm 0.16$	67.11
TABFACT	BERT	54.41 $\pm 0.51$	55.09	52.00 $\pm 0.96$	52.87	56.14 $\pm 0.45$	56.49	53.29 $\pm 1.17$	54.15
TABFACT	MASKLM	61.76 $\pm 1.06$	65.09	58.95 $\pm 0.64$	61.62	58.49 $\pm 0.15$	60.04	55.89 $\pm 0.40$	57.01
TABFACT	Inter	74.00 $\pm 0.32$	76.68	68.86 $\pm 0.48$	71.33	71.08 $\pm 0.78$	72.14	64.94 $\pm 0.34$	66.43
TABFACT	TABFACT	75.74 $\pm 0.18$	78.33	70.76 $\pm 0.55$	<b>72.95</b>	73.74 $\pm 0.95$	74.01	67.67 $\pm 0.96$	<b>68.03</b>

Table 2: Stage 1 and 2 ablation at 20,000 steps. majority, TABFACT (drop) and TABFACT (random) use majority voting (always predicting non-neutral), only the artificial data created by removing columns and only the random neutral statements respectively. All other models use both kinds of artificial statements.

respective column from the table. This procedure yields 651 unique new instances from the 4506 training examples. However, similarly to the first approach of pairing random statements and tables, the process is not perfect. It may happen that refuted statements continue to be refuted after removing some of the evidence, but in practice we find it beneficial to generate examples in this fashion.

The final training data is then created by taking the original crowdsourced training examples as positive examples and randomly sampling an equally-sized set of negative examples, where half of the negatives are random combinations of a statement with a table and the other half are drawn with replacement from the 651 artificial examples.

### 3.3 Entailment Stage

Training the entailment stage is rather straightforward, we train the model on the crowdsourced training data using the same hyper-parameters as Eisenschlos et al. (2020).

### 3.4 Calibration and Ensemble

As our training data for stage 1 is balanced but the development data is skewed we find it to improve accuracy if we trigger for examples with a logit larger than 4.0 (rather than 0.0). Empirically we also find the threshold of 4.0 to work better for the second stage. This could be explained by the fact that the development set has a different label distribution than the training set.

We train 5 models per stage and use them as an ensemble. The ensemble score is defined as the median of all the model scores. Using the median

worked better than the mean and voting in preliminary experiments.

## 4 Experimental Setup

In this section we explain the SEMTABFACT task and dataset and give additional details about the experimental setup we used.

The SEMTABFACT dataset consists of statements and tables from the scientific literature. It is much smaller than similar datasets such as TABFACT (Chen et al., 2020) and INFOTABS (Gupta et al., 2020). It is note-worthy that the training set only contains entailed and refuted statements while the dev and test set also contain neutral (unknown) statements. The statements were written by crowd workers, which presented with 7 different types of statements were instructed to write one statement of each type. The types of statements were using aggregation, superlatives, counting, comparatives, unique counting and the usage of the caption or common-sense knowledge.

The main metric of the task is the micro f1-score computed over the statements belonging to a table. The 3-way score takes all statements into account while the 2-way score is restricted to refuted and entailed statements.

## 5 Results

Table 2 compares our system to multiple baselines. Unless stated otherwise all baselines have been trained with the same neutral data generation as discussed above and for 20,000 steps. All numbers are based on 5 independent model runs. For all setups we report the median of the individual runs as

well as the results for a system based on the median logit of the 5 models. We report error margins for the medians as half the inter-quartile range.

Looking at the first stage of the system in Table 2, we see that the system based on TABFACT is the best choice for the initialization, out-performing a simple BERT model as well as models trained with only the mask-lm and intermediate pre-training on both dev and test ensemble accuracy. However, the model trained on the intermediate data gives higher median dev and test accuracy (e.g. 72.12 vs 70.76).

With respect to the data generation we observe that any kind of neutral data generation out-performs the majority baseline. Combining the column removal and random statements yields the best results. The drop in the 2-way metrics going from the majority Stage 1 model to a learned model is expected as that metric ignores all neutral statements in the eval set.

On the second stage of our system (Table 2), we see that a TAPAS model based on TABFACT outperforms the other baselines by a bigger margin than for Stage 1. For example, a model based on only MASKLM pre-training achieves 57.01 test f1 score while the TABFACT-based model achieves 68.03. We also found that for this stage there is a more pronounced difference between BERT and MASKLM (54.15 vs 57.01) and MASKLM and intermediate pre-training (57.01 vs 66.43).

Table 5 in the appendix shows the results for different number of steps and thresholds showing that results can be slightly tweaked by tuning them.

## 6 Analysis

Table 3 shows that the recall and precision on the *neutral* class are 37.6 and 71.4, respectively. Inspecting some instances of false positives, we find that the system is quite easily fooled; for example classifying the statement “*The lowest Factor 8 is 0.027*” as non-neutral for a table that has 5 columns labeled as Factor 1 to 5. False negatives are sometimes caused by failing to map words with typos (“paramters” vs “parameters”) or abbreviations (“measurement errors” vs “ME”). Adding harder examples of neutral statements to the training set could potentially further improve the identification. We also see that the recall on the refuted class (74.3) is lower than the recall of the entailed class (85.2) while their precision values are similar.

In Table 4 we construct mutually excluded groups of the validation set. Each set is identified

Reference \ Prediction	Non-neutral	Neutral	Recall
	Non-neutral	449	14
Neutral	58	35	37.6
Precision	88.6	71.4	

Reference \ Prediction	Refuted	Entailed	Recall
	Refuted	153	53
Entailed	36	207	85.2
Precision	81.0	79.6	

Table 3: Confusion matrix for Stage 1 and Stage 2 on the development set.

	Size	Acc	Baseline	ER
Overall	100.0	71.0	45.0	29.0
Superlatives	15.8	73.9	50.0	4.1
Aggregations	13.8	61.0	46.8	5.4
Comparatives	12.2	58.8	47.1	5.0
Negations	3.1	82.4	41.2	0.5
Multiple of the above	5.9	72.7	63.6	1.6
Other	49.1	75.1	43.6	12.2

Table 4: Accuracy and total error rate (ER) for different question groups derived from the same word heuristics defined in Eisenschlos et al. (2020). The baseline is simple class majority and the error rate in each group is taken with respect to the full set. Comparatives show the biggest margin for future improvements comparing with the overall system accuracy.

by specific keywords appearing in the statement, for example *Comparatives* must contain “higher”, “better”, “than”, etc. The full list is defined in the appendix of Eisenschlos et al. (2020). We observe that comparatives and aggregations have the largest total error rates, meaning that the biggest gains in overall accuracy can be made by improving those reasoning skills. Between these two, Comparatives have the lowest in-group accuracy. Table 6 and Table 7 in the appendix show the some analysis for Stage 1 and Stage 2, respectively. The trend for Stage 2 is similar to the overall trend whereas Stage 1 accuracy is relatively stable across the different groups except for comparatives where the accuracy drops from 87% overall to 81%.

Another class of examples with relatively low accuracy are statements around unique counting. We find that statements containing the word *different* have an accuracy of 51.3 (vs. 71% overall) and account for 3.4 percentage points of the total error rate. Examples include “*There are six different classes*” and “*They have ten different parameters*”.

## 7 Conclusion

We presented our contribution to the SEMTABFACT task (Wang et al., 2021) on table entailment. Our system consists of two stages that classify statements into non-neutral or neutral and refuted or entailed. Our model achieves 68.03 average micro f1-score on the test set. We showed that our procedure for creating artificial neutral statements improves the system over a majority baseline but results in a relatively low recall of 37.6. Other methods for creating harder neutral statements might further improve this value. In line with Eisenschlos et al. (2020), we find that pre-training on intermediate data improves the system accuracy over a system purely pre-trained with a MASKLM objective. While these initial results look promising, we find that the model struggles with statements that involve complex operations such as comparisons and unique counting.

## References

- Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. [Synthetic QA corpora generation with roundtrip consistency](#). In *Proceedings of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyong Zhou, and William Yang Wang. 2020. [Tabfact: A large-scale dataset for table-based fact verification](#). In *Proceedings of the International Conference on Learning Representations*.
- Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rationale, evaluation and approaches. *Journal of Natural Language Engineering*, 4.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. [Understanding tables with intermediate pre-training](#). In *Findings of the Association for Computational Linguistics: EMNLP*, pages 281–296, Online. Association for Computational Linguistics.
- Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. [Evaluating models’ local decision boundaries via contrast sets](#). In *Findings of the Association for Computational Linguistics: EMNLP*, pages 1307–1323, Online. Association for Computational Linguistics.
- Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. [Injecting numerical reasoning skills into language models](#). In *Proceedings of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.
- Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. [Infotabs: Inference on tables as semi-structured data](#). In *Proceedings of the Association for Computational Linguistics*, Seattle, Washington. Association for Computational Linguistics.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. [Don’t stop pretraining: Adapt language models to domains and tasks](#). In *Proceedings of the Association for Computational Linguistics*, Seattle, Washington. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.
- Jeremy Howard and Sebastian Ruder. 2018. [Universal language model fine-tuning for text classification](#). In *Proceedings of the Association for Computational Linguistics*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.
- Shankar Iyer, Nikhil Dandekar, , and Kornél Csernai. 2017. [Quora question pairs](#).
- Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. [Search-based neural structured learning for sequential question answering](#). In *Proceedings of the Association for Computational Linguistics*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.



- Saehan Jo, Immanuel Trummer, Weicheng Yu, Xuezhi Wang, Cong Yu, Daniel Liu, and Niyati Mehta. 2019. [Aggchecker: A fact-checking system for text summaries of relational data sets](#). *International Conference on Very Large Databases*, 12(12):1938–1941.
- Divyansh Kaushik, Eduard H. Hovy, and Zachary Chase Lipton. 2020. [Learning the difference that makes a difference with counterfactually-augmented data](#). In *Proceedings of the International Conference on Learning Representations*, Addis Ababa, Ethiopia.
- Rezka Leonandya, Dieuwke Hupkes, Elia Bruni, and Germán Kruszewski. 2019. [The fast and the flexible: Training neural networks to learn to follow instructions from small data](#). In *Proceedings of the International Conference on Computational Semantics*, pages 223–234, Gothenburg, Sweden. Association for Computational Linguistics.
- Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. [Unsupervised question answering by cloze translation](#). In *Proceedings of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.
- Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. [Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work?](#) In *Proceedings of the Association for Computational Linguistics*, Seattle, Washington. Association for Computational Linguistics.
- Felipe Salvatore, Marcelo Finger, and Roberto Hirata Jr. 2019. [A logical-based corpus for cross-lingual evaluation](#). In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 22–30, Hong Kong, China. Association for Computational Linguistics.
- Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. [A corpus of natural language for visual reasoning](#). In *Proceedings of the Association for Computational Linguistics*, pages 217–223, Vancouver, Canada. Association for Computational Linguistics.
- Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. [A corpus for reasoning about natural language grounded in photographs](#). In *Proceedings of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.
- Andreas Vlachos and Sebastian Riedel. 2015. [Identification and verification of simple claims about statistical properties](#). In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2596–2601, Lisbon, Portugal. Association for Computational Linguistics.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. [Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents \(sem-tab-facts\)](#). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Yushi Wang, Jonathan Berant, and Percy Liang. 2015. [Building a semantic parser overnight](#). In *Proceedings of the Association for Computational Linguistics*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.
- Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ikhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, Ugur Cetintemel, and Carsten Binnig. 2020. [Dbpal: A fully pluggable nl2sql training pipeline](#). In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '20*, page 2347–2361, New York, NY, USA. Association for Computing Machinery.
- Changxing Wu, Xiaodong Shi, Yidong Chen, Yanzhou Huang, and Jinsong Su. 2016. [Bilingually-constrained synthetic data for implicit discourse relation recognition](#). In *Proceedings of Empirical Methods in Natural Language Processing*, pages 2306–2312, Austin, Texas. Association for Computational Linguistics.
- Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. [Table fact verification with structure-aware transformer](#). In *Proceedings of Empirical Methods in Natural Language Processing*, pages 1624–1629, Online. Association for Computational Linguistics.

## Appendix

The appendix contains additional results and analysis tables.

### A Results

Table 5 shows the results for different number of steps and thresholds showing that results can be slightly tweaked by tuning them.

Steps	Thresh	f1 2-way		f1 3-way	
		Median	Ensemble	Median	Ensemble
20K	0.0	74.97 $\pm$ 0.35	76.41	69.96 $\pm$ 1.05	71.03
10K	4.0	<b>75.97</b> $\pm$ 1.48	76.47	70.68 $\pm$ 1.36	72.19
10K	0.0	75.84 $\pm$ 1.33	76.55	70.63 $\pm$ 1.21	72.27
20K	4.0	75.74 $\pm$ 0.18	<b>78.33</b>	<b>70.76</b> $\pm$ 0.55	<b>72.95</b>

Table 5: Ablation of steps and threshold on the dev set.

### B Analysis

Table 6 and Table 7 show the error rate contributions of different types of statements for Stage 1 and Stage 2, respectively. The trend for Stage 2 is similar to the overall trend (Table 4) whereas Stage 1 accuracy is relatively stable across the different groups except for comparatives where the accuracy drops from 87% overall to 81%.

	Size	Acc	Baseline	ER
<b>Overall</b>	100.0	87.1	83.3	12.9
<b>Superlatives</b>	15.8	90.9	89.8	1.4
<b>Aggregations</b>	13.8	88.3	87.0	1.6
<b>Comparatives</b>	12.2	80.9	79.4	2.3
<b>Negations</b>	3.1	88.2	64.7	0.4
<b>Multiple of the above</b>	5.9	93.9	87.9	0.4
<b>Other</b>	49.1	86.1	81.7	6.8

Table 6: Accuracy and total error rate (ER) for different question groups for Stage 1.

	Size	Acc	Baseline	ER
<b>Overall</b>	100.0	80.2	54.1	19.8
<b>Superlatives</b>	16.9	80.3	53.9	3.3
<b>Aggregations</b>	14.0	66.7	54.0	4.7
<b>Comparatives</b>	11.6	71.2	59.6	3.3
<b>Negations</b>	2.4	90.9	63.6	0.2
<b>Multiple of the above</b>	6.2	75.0	71.4	1.6
<b>Other</b>	48.8	86.3	53.0	6.7

Table 7: Accuracy and total error rate (ER) for different question groups for Stage 2.

# BOUN at SemEval-2021 Task 9: Text Augmentation Techniques for Fact Verification in Tabular Data

**Abdullatif Köksal**

Department of Computer Engineering  
Boğaziçi University  
abdullatif.koksal@boun.edu.tr

**Yusuf Yüksel**

Department of Computer Engineering  
Boğaziçi University  
yusuf.yuksel@boun.edu.tr

**Bekir Yıldırım**

Department of Computer Engineering  
Boğaziçi University  
bekir.yildirim@boun.edu.tr

**Arzucan Özgür**

Department of Computer Engineering  
Boğaziçi University  
arzucan.ozgur@boun.edu.tr

## Abstract

In this paper, we present our text augmentation based approach for the Table Statement Support Subtask (Phase A) of SemEval-2021 Task 9. We experiment with different text augmentation techniques such as back translation and synonym swapping using Word2Vec and WordNet. We show that text augmentation techniques lead to 2.5% improvement in F1 on the test set. Further, we investigate the impact of domain adaptation and joint learning on fact verification in tabular data by utilizing the SemTabFacts and TabFact datasets. We observe that joint learning improves the F1 scores on the SemTabFacts and TabFact test sets by 3.31% and 0.77%, respectively.

## 1 Introduction

Recognizing Textual Entailment (RTE) (Dagan et al., 2005) is one of the core NLP problems for understanding the semantic relations between words and sentences, which is useful for other tasks including Question Answering (Abacha and Demner-Fushman, 2019), Text Summarization (Lloret et al., 2008), and Text Classification (Yin et al., 2019). For the RTE task, datasets of various sizes (Dagan et al., 2005; Bowman et al., 2015) and from different domains (Romanov and Shivade, 2018) have been introduced. However, these works and datasets are solely focused on textual data without considering structured data such as tables.

Recently, question answering (Iyyer et al., 2017) and textual entailment datasets (Wenhu Chen and Wang, 2020; Wang et al., 2021) for tabular data have been introduced. SemEval-2021 Task 9 addresses the problem of statement verification

<sup>1</sup>The grammatical error exists in the given dataset.

*Distance statistics between buildings of ancient buildings and modern buildings to the main water channel (unit: meter).*

Index	Ancient building	Modern building
AVERAGE	174.095	273.917
STDEV.S	58.780	190.928
MIN	6.763	4.868
MAX	321.608	912.368
MEDIAN	173.010	243.885

Statement	Label
There are 2 types of building - Ancient building and Modern building.	Entailed
All the values of Ancient building is less than Modern building except MIN value.	Entailed
The value of Modern building is lesser than Ancient building in AVERAGE. <sup>1</sup>	Refuted

Figure 1: Sample table, description, and statements from SemTabFacts.

(Phase A) and evidence finding (Phase B) using tables from scientific articles (Wang et al., 2021). The shared task also introduced a new dataset, namely the SemTabFacts dataset, an example from which is provided in Figure 1. The goal of Phase A (Table Statement Verification) of the shared task is to determine whether a statement is entailed, refuted, or unknown given a table and its description (if available). For example, given the table and its description in Figure 1, the first two statements

are entailed, whereas the third statement is refuted. This example demonstrates that there are various challenges such as understanding numerical operations and comparisons as well as textual entailment.

Transformers architecture (Vaswani et al., 2017) enabled the pretraining of large language models, which achieve significant improvement in numerous NLP tasks (Wang et al., 2018, 2019). Recent works have also focused on pretraining language models for tabular data by introducing new embedding layers and objective functions, as well as large-scale augmented data to better represent numerical values and rankings (Herzig et al., 2020; Eisenschlos et al., 2020).

Data augmentation is a way to enrich training data to improve the supervised training scheme and is widely used in computer vision (Perez and Wang, 2017) and speech recognition (Park et al., 2019). Different text augmentation techniques such as back translation, synonym replacement, and text editing have been investigated for various tasks including text classification (Wei and Zou, 2019) and natural language inference (Min et al., 2020).

In this study, we aim at investigating the impact of text augmentation on the statement verification task from tables. We implement various text augmentation techniques based on WordNet (Miller, 1998), Word2Vec (Mikolov et al., 2013), and Back Translation (Yu et al., 2018) to enrich the statement variety in the SemTabFacts dataset. We finetune a recently introduced pretrained transformer architecture, the TAPAS model (Eisenschlos et al., 2020), for our approach. In addition, we investigate the domain adaptation and joint learning capabilities of two tabular fact verification datasets: SemTabFacts and TabFact. Promising results are achieved on the SemTabFacts test dataset.

## 2 Datasets

We use two different table-based fact verification datasets for the experiments: SemTabFacts (Wang et al., 2021) and TabFact (Wenhu Chen and Wang, 2020). We compare SemTabFacts and TabFact in terms of the average size of the tables, average word length of the statements, and the number of examples for each class in Table 1. We only report the statistics for the training sets, since the development and test sets have similar distributions with the training sets in both datasets. There is almost an order of magnitude difference between the datasets in terms of the number of tables and

	SemTabFacts	TabFact
# Tables	981	13,182
# Statements	4,506	92,283
# Entailed	2,818	50,820
# Refuted	1,688	41,463
Avg. Row Size	9.0±8.0	13.5±8.6
Avg. Column Size	5.3±2.9	6.4±1.7
Avg. Statement Length (Words)	11.5±7.1	13.2±4.5

Table 1: Comparative statistics of SemTabFacts and TabFact.

statements. Furthermore, we observe that the average table size and average statement length in terms of words are greater in TabFact than SemTabFacts.

**SemTabFacts** (Wang et al., 2021): This dataset consists of tables from articles published in Elsevier, which are available on ScienceDirect. After filtering complicated examples, five entailed and five refuted statements about these tables are generated by high-quality crowd-sourcing. These statements are further verified by additional crowd-source workers, especially for filtering out ungrammatical sentences. To increase the quality level, Wang et al. (2021) further verified the statements in the development and test sets. The SemTabFacts dataset also contains automatically generated statements and unknown classes in the development and test sets for the fact verification and evidence finding tasks. In this study, we target two-way (Entailed / Refuted) classification without automatically generated statements for the fact verification task.

SemTabFacts releases tables and statements in XML format. We convert these tables into CSV format to properly use in our models. Due to cells with multirow and multicolumn features in XML, we could not accurately convert all tables into CSV, which might affect our models’ overall performance. We manually checked the XML to CSV conversion of 50 tables. We identified three errors related to multirow and multicolumn features, and one error that causes a missing column.

**TabFact** (Wenhu Chen and Wang, 2020): This dataset crawls tables from Wikipedia articles following previous works on table question answering (Pasupat and Liang, 2015; Zhong et al., 2017).

Designed input parameters during experimental set-up.

Experimental bottle ID	Material type	Material amount, g
A	Inoculum	200.2
B	Inoculum	200.4
1A	Inoculum + Olive cake	200.4
1B	Inoculum + Olive cake	199.7

Augmentation Methods	Sentence
Original	199.7 is the lowest Inoculum compare to all others.
WordNet	199.7 is the small Inoculum compare to all others.
Word2Vec	199.7 is the lowest Inoculum comparisons to all others.
Back Translation	199.7, compared to others is the most low inoculum.

Figure 2: Generated sentences by different text augmentation methods for the same statement. The table for the original statement is given above with some modifications.

Complicated tables including multirows, multi-columns, and latex symbols, and large tables with more than 50 rows or 10 columns were filtered out. Amazon Mechanical Turk was used to generate simple and complex statements about tables. The Mechanical Turk workers also filtered out poor statements that have grammatical errors or vague claims. Finally, annotator agreement scores were computed by having the same set of statements labeled by another set of Mechanical Turk workers.

### 3 Methods

#### 3.1 TAPAS

Deep transformers models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have achieved significant improvement in different NLP tasks as seen in the GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) benchmarks. However, it is not straightforward to benefit from these models for structured data formats such as tables or graphs. TAPAS (Herzig et al., 2020) introduces different objectives such as cell selection and aggregation prediction, and new additional embeddings such as column/row id and rank id over BERT’s architecture, which are more suitable for complex numerical operations and comparisons in tables. The TAPAS model has been designed by focusing on the task of question answering over tables (Herzig et al., 2020). However, TAPAS fails to handle complex compositional structures like multiple aggregations and large tables due to the maximum length limit of the tokenizer.

To overcome the problems in (Herzig et al., 2020), recently, Eisenschlos et al. (2020) introduced new mechanisms such as table pruning to

make TAPAS work with large tables without memory errors. Furthermore, two augmentation methods for statements were presented (Eisenschlos et al., 2020). The first one is based on creating counterfactual statements by replacing entity mentions with other entities on entailed examples to populate negative samples. The second one is based on a synthetic data generation method to populate statements with complex numerical operations.

In this study, we use a TAPAS model from HuggingFace’s Transformers library (Wolf et al., 2019). This model is pretrained on Masked Language Model and additional intermediate pretraining steps as discussed in (Eisenschlos et al., 2020). In addition, it is finetuned on the TabFact dataset (Wenhu Chen and Wang, 2020). We further finetuned this model on SemTabFacts (Wang et al., 2021) with additional augmentation steps by utilizing WordNet, Word2Vec, and back translation.

#### 3.2 Text Augmentation

##### 3.2.1 WordNet

WordNet (Miller, 1998) is a lexical database that groups words into adverbs, adjectives, nouns, and verbs, and shows the relations between them such as hyponymy, antonymy, and synonymy. In this work, we focus on swap-based WordNet augmentation that changes words by their WordNet synonyms. The implementation is done by the TextAttack (Morris et al., 2020) library. As shown in Figure 2, the word *lowest* is changed to *small* by synonym swapping.

##### 3.2.2 Word2Vec

Word2Vec (Mikolov et al., 2013) is a technique to find dense word embeddings by shallow networks.

Training Set	Dev	Test
SemTabFacts	0.7661	0.7044
SemTabFacts+WN	0.7791	<b>0.7294</b>
SemTabFacts+W2V	0.7486	0.7201
SemTabFacts+BT	0.7725	0.6941
SemTabFacts+WN+W2V	0.7648	0.7147
SemTabFacts+WN+BT	<b>0.7869</b>	0.7217
SemTabFacts+W2V+BT	0.7614	0.7202
SemTabFacts+W2V+WN+BT	0.7484	0.7101

Table 2: F1 scores of different augmentation techniques on SemTabFacts. WN, W2V, and BT represent WordNet, Word2Vec, and Back Translation, respectively.

It helps to represent syntactic and semantic features of words by a dense vector. Due to the low dimensional space, similar words and synonyms have closer word embeddings. We make use of this feature of Word2Vec to replace words with their Word2Vec synonyms by TextAttack library. For example, this augmentation technique changes the word *compare* to *comparisons* as shown in Figure 2. While WordNet augmentation preserves the part of speech tags of the words, Word2Vec augmentation may distort the part of speech tags and may produce ungrammatical sentences.

### 3.2.3 Back Translation

The back translation technique paraphrases a given sentence from a source language by translating it into another target language and then translates it back into the source language. It was first introduced as a data augmentation mechanism for the reading comprehension task (Yu et al., 2018), where significant improvement was observed by back translation augmentation. Recent machine translation systems (Sennrich et al., 2016) are robust to back translation mechanism and tend to produce the same sentence. To overcome this issue, we used two different versions of the same system. First, we translated the statements in English to Turkish by Google Translate<sup>2</sup>. Then, we translated the Turkish statements into English by the GOOGLETRANSLATE function in Google Sheets. The back translation method paraphrases the sentence and unlike the WordNet and Word2Vec approaches, it may change word order in addition to the words, as illustrated in Figure 2.

<sup>2</sup><https://translate.google.com>

Training Set	SemTabFacts		TabFact	
	Dev	Test	Dev	Test
SemTabFacts	0.7661	0.7044	0.7435	0.7471
TabFact	0.7284	0.7019	<b>0.8200</b>	0.8178
SemTabFacts+TabFact	<b>0.7992</b>	<b>0.7335</b>	0.8167	<b>0.8255</b>

Table 3: F1 scores of domain adaptation and joint learning capabilities of SemTabFacts and TabFact.

## 4 Experimentation and Results

We conduct two different experimental setups to compare our results. In both experiments, we finetune all layers of a pretrained TAPAS model and its classifier head. First, we finetune the TAPAS model on the SemTabFacts dataset with all combinations of different augmentation techniques. Second, instead of using augmentation techniques, we finetune the TAPAS model on TabFact only and then on SemTabFacts and TabFact jointly and compare the results on the test sets of TabFact and SemTabFacts. In all these experiments, we use the AdamW (Loshchilov and Hutter, 2018) optimizer with a  $5e-5$  learning rate and 0.01 weight decay. We set batch size as 8 with 2 accumulation steps, and the number of steps used for linear warm-up is 100 in SemTabFacts training and 2000 in TabFact and joint training. We finetune this model over 10 epochs and decide the best model based on the development set. We use the official evaluation metric, which is the macro-average of F1 scores over the tables.

In the augmentation steps, we include new augmented statements for each statement and augmentation method to the training data of SemTabFacts. The original versions of the development and test sets are used without any augmentation. We observe that different augmentation techniques in SemTabFacts can improve F1 scores on the test set as shown in Table 2. The best model for the development set of SemTabFacts is the model with WordNet and back translation augmentations. Besides, all augmentation techniques, except back translation, improve the test F1 score over the base model without augmentation. Finally, we observe that WordNet augmentation increases the test F1 score by 2.5% over the base model without augmentation.

In Table 3, we investigate the domain adaptation and joint learning capabilities of SemTabFacts and

TabFact. We have finetuned three separate models, with SemTabFacts training data, TabFact training data, and SemTabFacts and TabFact training data. We evaluate these finetuned models on the development and test sets of the datasets. The original versions of the training, development, and test sets are used in these experiments without any additional augmentation. The model trained with TabFact and SemTabFacts data achieved the highest F1 scores on the test sets of both datasets. The joint model improves the F1 score by 3.31% and 0.77% on the SemTabFacts and TabFact test sets, respectively. Further, we observe that we can achieve similar scores on the SemTabFacts test set when the model is trained on the SemTabFacts training data or on the TabFact training data.

We further analyzed the errors in terms of table sizes (number of rows x number of columns) and length of the statements. However, our results indicate that there is no significant difference in F1 scores for different table sizes and different lengths of statements. Our models have similar performance for small and large tables as well as for short and long statements.

## 5 Conclusion

In this work, we described our models for the Table Statement Support Subtask (Phase A) of SemEval-2021 Task 9. Our base model relies on the recently introduced pretrained transformer architecture for tabular data, TAPAS. We proposed three different augmentation techniques which are based on WordNet, Word2Vec, and Back Translation. We showed that all combinations of these augmentation techniques except Back Translation perform better on the test set than methods without augmentation. Furthermore, we investigated the domain adaptation and joint learning capabilities of SemTabFacts and TabFact. We showed that our best model in terms of development and test F1 for SemTabFacts occurs when we trained TAPAS jointly on the SemTabFacts and TabFact datasets. Additionally, we illustrated that the joint model achieves better results on the TabFact test set than the model trained only on the TabFact training dataset. As future work, we plan to focus on better preprocessing the SemTabFacts dataset and more diverse augmentation techniques by integrating perplexity scores of augmented statements.

## Acknowledgments

TUBITAK-BIDEB 2211-A Scholarship Program (to A.K.), and TUBA-GEBIP Award of the Turkish Science Academy (to A.O.) are gratefully acknowledged.

## References

- Asma Ben Abacha and Dina Demner-Fushman. 2019. A question-entailment approach to question answering. *BMC bioinformatics*, 20(1):1–23.
- Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. **Understanding tables with intermediate pre-training**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. **TaPas: Weakly supervised table parsing via pre-training**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.
- Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. **Search-based neural structured learning for sequential question answering**. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

- Elena Lloret, Oscar Ferrández, Rafael Munoz, and Manuel Palomar. 2008. A text summarization approach under the influence of textual entailment. In *NLPCS*, pages 22–31.
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, volume 26, pages 3111–3119.
- George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.
- Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020. [Syntactic data augmentation increases robustness to inference heuristics](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online. Association for Computational Linguistics.
- John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. [TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.
- Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. 2019. Specaugment: A simple data augmentation method for automatic speech recognition. *Proc. Interspeech 2019*, pages 2613–2617.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Luis Perez and Jason Wang. 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.
- Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. SemEval-2021 Task 9: A fact verification and evidence finding dataset for tabular data in scientific documents (SEM-TAB-FACTS). In *Proceedings of SemEval*.
- Jason Wei and Kai Zou. 2019. [EDA: Easy data augmentation techniques for boosting performance on text classification tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.
- Jianshu Chen Yunkai Zhang Hong Wang Shiyang Li Xiyu Zhou Wenhua Chen, Hongmin Wang and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.
- Wenpeng Yin, Jamaal Hay, and Dan Roth. 2019. [Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3914–3923, Hong Kong, China. Association for Computational Linguistics.
- Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V



Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *International Conference on Learning Representations*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

# IITK at SemEval-2021 Task 10: Source-Free Unsupervised Domain Adaptation using Class Prototypes

Harshit Kumar\*    Jinang Shah\*    Nidhi Hegde\*  
Priyanshu Gupta\*    Vaibhav Jindal\*  
Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{hakumar, sjinang, nidhih, guptap, vaibhavj}@iitk.ac.in  
ashutoshm@cse.iitk.ac.in

## Abstract

Recent progress in deep learning has primarily been fueled by the availability of large amounts of annotated data that is obtained from highly expensive manual annotating processes. To tackle this issue of availability of annotated data, a lot of research has been done on unsupervised domain adaptation that tries to generate systems for an unlabelled target domain data, given labelled source domain data. However, the availability of annotated or labelled source domain dataset can't always be guaranteed because of data-privacy issues. This is especially the case with medical data, as it may contain sensitive information of the patients. Source-free domain adaptation (SFDA) aims to resolve this issue by using models trained on the source data instead of using the original annotated source data. In this work, we try to build SFDA systems for semantic processing by specifically focusing on the negation detection subtask of the SemEval 2021 Task 10. We propose two approaches - *ProtoAUG* and *Adapt-ProtoAUG* that use the idea of self-entropy to choose reliable and high confidence samples, which are then used for data augmentation and subsequent training of the models. Our methods report an improvement of up to 7% in F1 score over the baseline for the Negation Detection subtask.

## 1 Introduction

The availability of large scale datasets has been the main driving factor behind the success of supervised deep learning in the recent times. However, the process of data annotation is very expensive and time consuming, being one of the major challenges in extending deep learning techniques for new tasks.

One possible way to solve this problem is to *train* a machine learning model using an annotated

\* Authors contributed equally to the work. Names in alphabetical order.

source dataset to assist the annotation process over some unlabelled target dataset. However, there may be differences in the source and target domain distributions, which may lead to inaccuracies. Thus the challenge is to update the weights of the source classifier to generalize it well on the target domain. This aligns with the well studied problem of *Unsupervised Domain Adaptation* (UDA) (Kouw and Loog, 2019; Wang and Deng, 2018; Ramponi and Plank, 2020)

A common denominator across many popular UDA methods is their dependence on large amounts of labelled source domain data (Ganin and Lempitsky, 2015; Saito et al., 2018). However, many a times it is not possible to release the source domain dataset because of privacy concerns. This problem becomes particularly relevant when working with clinical Natural Language Processing (NLP) datasets because they contain highly sensitive information which cannot be freely distributed. To tackle these data sharing constraints, the framework of *Source-Free Domain Adaptation* (SFDA) is gaining interest (Laparra et al., 2020). In SFDA, instead of sharing the source domain data, only a model that has been trained on the source domain data is shared. This model is then used for solving the original task for the unlabelled target domain.

SemEval 2021 Task 10 (Laparra et al., 2021) asks participants to develop SFDA models for two subtasks. The first subtask involves **Negation Detection**, where we are required to determine whether or not a clinical entity (*diseases, symptoms, etc.*) mentioned in a sentence is negated in the given context. The second subtask is of **Time Expression Recognition**, where the objective is to detect and label all time expressions mentioned in a given document. In this work we have focused on the negation subtask. For solving this subtask our strategy is to make use of high-confidence prototypes from the *target* domain to reinforce the *target*-

*specific* features of the source model. We propose a simple augmentation technique that makes use of these *high-confidence* prototypes to generate labelled artificial datapoints. These augmented samples are then used to perform supervised fine-tuning of our source model. Using our methods, we were able to obtain upto a 7% improvement in F1 score over the baseline. The code for models and experiments is made available via GitHub.<sup>1</sup>

## 2 Background

In source free domain adaptation (SFDA) problem for semantic analysis, the goal is to create accurate systems for un-annotated target domain data. For these tasks, we are provided with un-annotated target domain data, and a model trained on the annotated source domain data for a similar task.

The shared task is further divided into 2 sub-tasks – Negation detection and Time expression recognition. In this work we focus only on the first subtask.

**Negation Detection:** The task is to classify clinical event mentions for whether they are negated by their context. This is essentially a “span-in-context” classification problem, where both the entity to be classified and its surrounding context are to be considered. For example, the sentence - “*Has no <e>diarrhea </e>and no new lumps or masses*” has the entity *diarrhea* which is negated by its context, and the model’s task is to correctly identify this entity as negated.

**Pretrained Models:** For negation detection, a given pre-trained classification model has been fine-tuned on the 10,259 instances in the SHARP Seed dataset(Rea et al., 2012) of de-identified clinical notes from Mayo Clinic of which 902 instances are negated.

**Practice Data:** The development data for the negation task is a subset of the i2b2 2010 Challenge (Uzuner et al., 2011) on concepts, assertions, and relations in clinical text. The practice dataset is further divided into *train* and *dev* splits. The *train* split contains 2886 unlabeled sentences while the *dev* split is composed of 5545 labeled sentences.

**Test Data:** A part of the MIMIC III corpus v1.4 (Johnson et al., 2016), is used as the test set for the negation detection subtask. The processed test data contains around 600K instances.

<sup>1</sup><https://github.com/purug2000/protoAug.git>

## 2.1 Prior Work

The limitations in creating large scale annotated datasets have led to a large amount of work on unsupervised domain adaptation in the recent years (Ganin and Lempitsky, 2015; Ganin et al., 2016; Tzeng et al., 2017; Saito et al., 2018). However, most of this work assumes free availability of source domain data. In source free domain adaptation (SFDA) problems, when no annotated source data is available, and only a pretrained model is provided, the domain adaptation problem becomes rather difficult, and this remains a largely unexplored area in the NLP community. However, there have been some recent works in the computer vision domain that attempt to solve this problem. Hou and Zheng (2020) propose a model to transfer the style of source images to that of the target images by exploiting the information stored in the batch normalization layers of the pre-trained model. In another work, (Kim et al., 2020) observed that the target domain data points with lower entropy are generally classified correctly and are reliable enough to generate pseudo labels for the entire target dataset.

The two sub-tracks for the current SemEval task are well studied problems in the supervised setting and a lot of work has been done on developing models for both the negation detection in clinical settings (Chapman et al., 2001; Cotik et al., 2016) and the time expression recognition task (Laparra et al., 2018). However, in this work, we attempt to approach the negation detection task from the perspective of SFDA, and not on improving these techniques in general.

## 3 System Overview

In this paper we offer a novel perspective on the problem of domain adaptation for the negation detection task in clinical NLP. The proposed approaches attempt to utilize some of the aspects of both self-learning and semi-supervised learning, as explained next.

**Class Prototypes:** If there was any access to the labeled target data then the most intuitive approach would have been the fine-tuning. But for unlabeled case, in order to fine-tune the pre-trained network  $S$ , it would become necessary to generate a labeled set of data from the given unlabeled target domain data. One way to approach this would be through a concept from self-learning, i.e., by finding the most reliable samples from the target data over which the

model  $S$  is sufficiently confident and using these predictions as the corresponding ground truth. In order to find reliable target samples, self-entropy  $H$  can be used to quantify the prediction uncertainty:

$$H(x) = - \sum_{k=1}^K p_k^S(x) \log(p_k^S(x)) \quad (1)$$

Here,  $S$  refers to the network (pre-trained classifier),  $K$  are total number of classes, and  $p_k^S(x)$  is the probability of the instance  $x$  belonging to the class  $k$ .

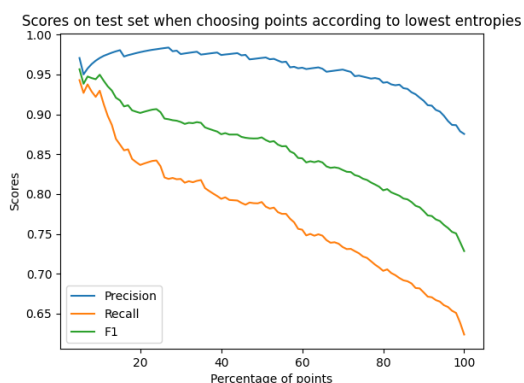


Figure 1: For any point P on the x-axis, the y-axis shows the performance scores on the data points having their self-entropies in the lowest P percentile.

The samples with smaller self-entropy indicate that the classifier is more confident over them, and these are referred to as *Prototypes*. The relationship between the self-entropy value and certainty in the prediction can be clearly observed in figure 1. Due to very high confidence of the model and accuracy over these prototypes, we can safely consider their predicted labels as their actual true labels. Here, one crucial hyperparameter to consider is the self-entropy threshold below which the residing target samples will be identified as prototypes. The key issue faced while determining the value of this hyperparameter is the disparities and highly imbalanced data distribution between the classes. Especially, in the negation task, the presence of the negated sentences in the practice data itself is in a very small proportion than that of non-negated sentences making negated a minority and non-negated a majority class. Analysis of the practice data (figure 2) further showed that the lowest self-entropy achieved by the negated class is far higher than that of non-negated class. For now, the self-entropy threshold value is defined by the 50th percentile (median) self-entropy value of the

minority class i.e. negated class. The rationale for this is provided in the Analysis section (section 5).

Unfortunately, the fine-tuning of the pre-trained network  $S$  with the prototypes identified from the target samples, didn't enable the trained network to generalise its performance over unseen target data as it made it highly likely to overfit on the prototypes without any early intervention. With this approach, we were able to get improvement of about 1% in F1 score over the baseline but that too with the highly unstable and unreproducible training results. Only using the reliable samples became a major issue as the trained model seemed to be unaware of the samples over which the model was less confident before.

**Augmentation:** To address the issue of lack of generalisation of the model towards the unseen and not-so-reliable target data, we propose the use of **augmentation**, inspired by the Mean Teacher model proposed by Tarvainen and Valpola (2018). The basic idea is to regularize the network by introducing noise to the prototypes in a way that can keep the label same and thereby creating new data points with the known labels. Mean Teacher model uses similar strategy for the labeled data points, instead here we apply that to the identified prototypes from the target data. This way, the pre-trained network can be subjected to a further constructive training by introducing a set of new labelled samples, which could help with the generalisation for the trained model.

Another use of the augmentation here is to address the issue of highly skewed and imbalanced distribution of data between the classes. Here, the augmentation is utilized not just to generate new samples to regularize the network but also to make the data distribution balanced across the classes by adding new samples in accordance with the preferred proportions.

Although there could be many possible ways to augment the samples so that their labels can be maintained, for this specific task, since we have a highlighted concept term in each sentence, we have used the augmentation by replacing that concept term. The new sentences are generated from their parent prototype sentences by replacing the concept term with a concept term from any randomly selected sentence of the same class in the data set. Here we have assumed that most of the concept terms will represent medical condition and thus nouns. Even when the grammar and the sentence

structure is preserved, the result of the augmentation might still not be perfect due to the possible ambiguity in the concept term selection or incorrect grammar in the original sentence itself. For example, consider the sentence “...<e> cortisol </e> is currently pending.”. Here it is not clear if the entity “cortisol” is negated or non-negated. However, it should be noted that such ambiguities are relatively less frequent which in turn implies that the augmentation will suffice its purpose for the majority of the time.

**ProtoAUG and Adapt-ProtoAUG:** Combining the concepts discussed above, here we propose our two approaches *ProtoAUG* and *Adapt-ProtoAUG*, both of which share the same set of concepts of class prototypes and augmentation as explained before. For both *ProtoAUG* and *Adapt-ProtoAUG*, the common underlying procedure is as follows: first the prototypes are identified from the target domain data and then the augmentation follows with the intent of regularizing the network and to create a more balanced distribution of samples across the classes. Now both the prototypes and their augmented samples with their respective labels are used with cross entropy loss to update the weights of the feature extractor module  $F$  of the pre-trained network  $S$ , with classifier module  $C$  of the pre-trained network being frozen.

The fundamental difference between *ProtoAUG* and *Adapt-ProtoAUG* is about the adaptive nature of *Adapt-ProtoAUG* in recognizing the prototypes from the original target domain dataset after every epoch, which *ProtoAUG* does only at the beginning of the training. *Adapt-ProtoAUG* makes incremental changes to the percentile score (initially 50) for the self-entropy threshold. The intuition behind using this strategy is that as the training proceeds, model will become more confident on the training samples and the entropy values for all the samples will significantly decrease. A possible drawback of using a fixed percentile criteria for the threshold at every epoch would likely exclude some reliable samples even when they achieve objectively quite lower values of self-entropy. To avoid such scenarios, apart from repeating the same process of prototype identification followed by augmentation after every epoch, we also propose to increase the percentile score for determining the self-entropy threshold in an uniform manner throughout the training with some fixed upper bound (70). This is also explained further in the Analysis (section 5).

## 4 Experimental Setup

**Model:** The pre-trained model used in subtask-1 is a RoBERTa (Liu et al., 2019) based sequence classification model<sup>2</sup> provided by organizers after training on source domain data inaccessible to us. It has two modules - a feature extractor and a classifier that operates over the output [CLS] token from the feature extractor.

**Data:** In the practice phase, *train* split of the practice dataset was used to further train the pre-trained model whereas the *dev* split was used as a validation set for the hyper-parameter tuning. In the evaluation phase, due to our computational constraints, we were only able to utilize a randomly selected subset of 25k samples from around 600k sentences of the test dataset, for retraining of the pre-trained model. For evaluation the organisers use an annotated subset of the test dataset. During the evaluation phase this subset was kept hidden from the participants.

**Hyper-parameters setting:** For *ProtoAug*, self-entropy percentile threshold is set to 50% whereas as for *Adapt-ProtoAUG* it is uniformly increased after every epoch from being 50% at the first to being 70% at the final epoch. Using augmentation, the final number of samples per class is set to be  $x$  times the number of prototypes belonging to the majority (non-negated) class. In our experiments, we choose  $x$  to be 4. For both the approaches, the model training is performed for 10 epochs. During the test phase, we reuse the hyper-parameters obtained from the practice phase. Further details about hyper-parameter selection can be found in Appendix A.

## 5 Results

Table 1 shows results on the development data for the two approaches - *ProtoAUG* and *Adapt-ProtoAUG*. For reference, results obtained from the pre-trained model are shown as the baseline.

Model	F1 score	Precision	Recall
Baseline	0.834	0.850	0.818
ProtoAUG	0.877	0.948	0.816
Adapt-ProtoAUG	<b>0.888</b>	<b>0.959</b>	<b>0.827</b>

Table 1: Results obtained on dev data of practice phase

<sup>2</sup>Model is available on [https://huggingface.co/tmills/roberta\\_sfda\\_sharpseed](https://huggingface.co/tmills/roberta_sfda_sharpseed)

Table 2 shows results of our two proposed approaches with the baseline model on the test set. In evaluation phase, the observed improvement in F1-score was of roughly 7% from the original baseline.

Model	F1 score	Precision	Recall
Baseline	0.66	0.917	0.516
ProtoAUG	0.706	<b>0.939</b>	0.566
Adapt-ProtoAUG	<b>0.729</b>	0.876	<b>0.624</b>

Table 2: Results obtained on test data of evaluation phase

**Analysis:** To ascertain the relationship between self-entropy and a prediction’s reliability, we analyse the baseline performance scores for the data points within the varying self-entropy percentile threshold as shown in figure 1. For the baseline, we observe a direct correlation between a lower self-entropy and a higher prediction score on the respective data points. This further supports the use of low self-entropy data points as class prototypes in our proposed approaches.

As shown in figure 2, the lowest self-entropy achieved by minority (negated; label 1) class is far higher than that of majority (non-negated; label -1) class. This may be attributed to the skewed nature of the target dataset and potentially the source dataset as well.

Another interesting observation from figure 2 is that most of the majority class samples have a self-entropy lower than the lowest self-entropy achieved by the minority class. Thus, for selecting the threshold for prototype selection, we apply the percentile-based criteria only on the self-entropy values of the minority class.

For prototype selection, instead of using an absolute threshold value, we have chosen a percentile-based entropy threshold as it adapts relatively well across different domains. This follows from the fact that confidence of the model may vary from domain to domain due to which a threshold chosen for one domain might not be a good criteria for another domain.

For Adapt-ProtoAUG, as the upper bound for the self-entropy threshold is increased beyond 70, we observed a gradual decline in the model’s performance for the dev set. This may be due to the 85% precision of the baseline. Precision here refers to the proportion of correctly classified negated

samples to the model’s total number of negated predictions. So, it could be the case that as the threshold reaches near or get past the precision score, the probability of identifying a wrongly labelled sample as a prototype will rapidly increase. Compared to ProtoAUG, as the percentile threshold was increased from 50 to 70, we observed an overall increment of recall for both the dev and test dataset. Furthermore, introducing augmentation in the framework drastically increased the stability and reproducibility of the training process.

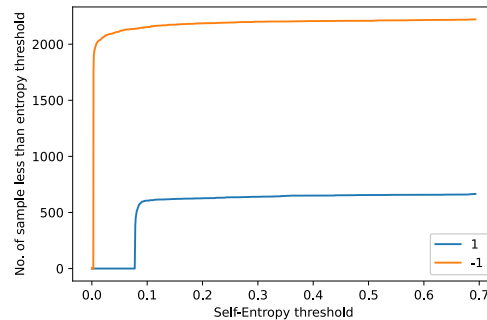


Figure 2: Cumulative number of sample vs entropy threshold curve

Appendix B provides some insights from t-SNE analysis, visually justifying the performance improvement. It analyses the predictions of the baseline model and Adapt-ProtoAUG over a fixed two-dimensional feature-space, comparing the similarities of their respective predictions with the original ground-truth labels. We observe that Adapt-ProtoAUG can capture the label distribution better than the baselines by performing well on various non-trivial data-points.

## 6 Conclusion

In this work, we carefully explored the problem of source-free domain adaptation for the Negation Detection subtask. We studied the importance of the confidence that a model places on its prediction and analyzed its formulation in terms of the samples’ self-entropy scores. Further, using those insights, we proposed two simple and intuitive approaches, namely ProtoAUG and Adapt-ProtoAUG for the Negation Detection Subtask and got an improvement of 7% on the test set with respect to the baseline model.

## References

- Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. 2001. [A simple algorithm for identifying negated findings and diseases in discharge summaries](#). *Journal of Biomedical Informatics*, 34(5):301 – 310.
- Viviana Cotik, Roland Roller, Feiyu Xu, Hans Uszkoreit, Klemens Budde, and Danilo Schmidt. 2016. [Negation detection in clinical reports written in German](#). In *Proceedings of the Fifth Workshop on Building and Evaluating Resources for Biomedical Text Mining (BioTxtM2016)*, pages 115–124, Osaka, Japan. The COLING 2016 Organizing Committee.
- Yaroslav Ganin and Victor Lempitsky. 2015. [Unsupervised domain adaptation by backpropagation](#). volume 37 of *Proceedings of Machine Learning Research*, pages 1180–1189, Lille, France. PMLR.
- Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. 2016. [Domain-adversarial training of neural networks](#). *Journal of Machine Learning Research*, 17(59):1–35.
- Yunzhong Hou and Liang Zheng. 2020. [Source free domain adaptation with image translation](#).
- Alistair E.W. Johnson, Tom J. Pollard, Lu Shen, Liwei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. [Mimic-iii, a freely accessible critical care database](#). *Scientific Data*, 3(1).
- Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyungseob Park, and Priyadarshini Panda. 2020. [Domain adaptation without source data](#).
- Wouter M. Kouw and Marco Loog. 2019. [An introduction to domain adaptation and transfer learning](#).
- Egoitz Laparra, Steven Bethard, and Timothy A Miller. 2020. [Rethinking domain adaptation for machine learning over clinical language](#). *JAMIA Open*, 3(2):146–150.
- Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. [SemEval 2018 task 6: Parsing time normalizations](#). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 88–96, New Orleans, Louisiana. Association for Computational Linguistics.
- Egoitz Laparra, Yiyun Zhao, Steven Bethard, and zlem Uzuner. 2021. [SemEval 2021 Task 10 - Source-Free Domain Adaptation for Semantic Processing\(to appear\)](#). *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).
- Alan Ramponi and Barbara Plank. 2020. [Neural unsupervised domain adaptation in nlp—a survey](#).
- Susan Rea, Jyotishman Pathak, Guergana Savova, Thomas A. Oniki, Les Westberg, Calvin E. Beebe, Cui Tao, Craig G. Parker, Peter J. Haug, Stanley M. Huff, and Christopher G. Chute. 2012. [Building a robust, scalable and standards-driven infrastructure for secondary use of ehr data: The sharpn project](#). *Journal of Biomedical Informatics*, 45(4):763–771.
- Translating Standards into Practice: Experiences and Lessons Learned in Biomedicine and Health Care.
- Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. [Maximum classifier discrepancy for unsupervised domain adaptation](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Antti Tarvainen and Harri Valpola. 2018. [Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results](#).
- Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. [Adversarial discriminative domain adaptation](#). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- zlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. [2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text](#). *Journal of the American Medical Informatics Association*, 18(5):552–556.
- Mei Wang and Weihong Deng. 2018. [Deep visual domain adaptation: A survey](#). *Neurocomputing*, 312:135–153.

## A Training Hyper-parameters

We train our models with a batch size of 32 using SGD optimizer with initial  $lr = 0.0005$  which decays after every iteration by multiplying it with  $(1 + 10 * itr / max\_iter)^{-0.75}$  (where  $itr$  is current iteration and  $max\_iter$  is maximum iteration of training),  $weight\_decay = 0.0005$  and  $momentum = 0.9$ .

## B t-SNE Analysis

We performed low dimensional analysis of the models using tsne. In the following figures, we took the 768 dimensional output of the baseline roberta model for the test dataset, and projected it in two dimensions using tsne.

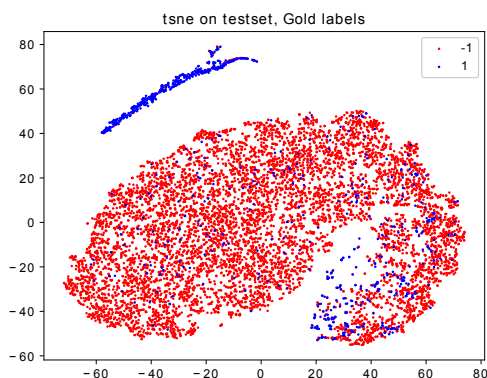


Figure 3: Ground truth labels. Blue points correspond to negation = 1.

- Figure 3 shows the the 2-dimensional tsne plots with the original ground truth labels. The blue points are the true positives and reds are the true negative points. We broadly observe two clusters - a smaller one with majority of points being true positives and a larger cluster with a majority of points being negatives. We also observe some blue points scattered in the red cluster and vice-versa.
- Figure 4 shows the predictions of the baseline model on the target domain. We see that the baseline classifier segregates the test data into almost perfect clusters, and thus misclassifies the scattered points.
- Figure 5 shows the prediction results of adapt-protoaug. In this case the F1 score improved from the baseline score by around 7%. Looking at the figure, we clearly see an improvement

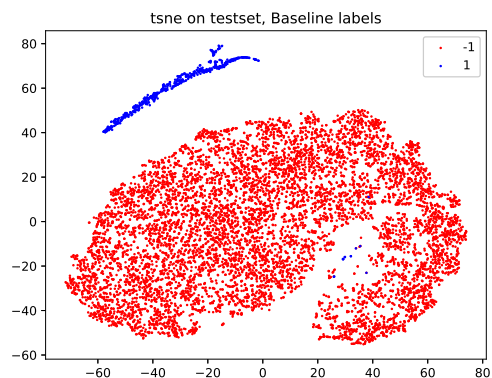


Figure 4: The predictions of the baseline model on the test set of the target domain.

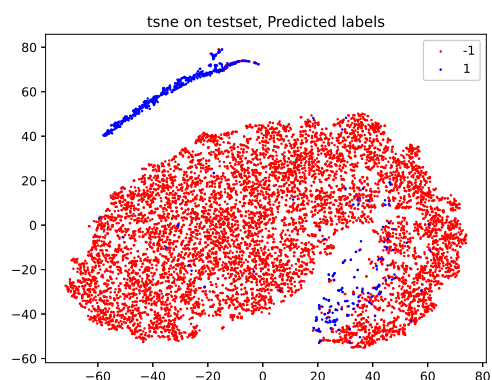


Figure 5: Visualisation of the predictions of an improved model

with respect to the baseline model, as we are now able to correctly capture some of the points that randomly fall within in the opposite cluster.



# PTST-UoM at SemEval-2021 Task 10: Parsimonious Transfer for Sequence Tagging

Kemal Kurniawan<sup>1</sup> Lea Frermann<sup>1</sup> Philip Schulz<sup>2\*</sup> Trevor Cohn<sup>1</sup>

<sup>1</sup>School of Computing and Information Systems, University of Melbourne

<sup>2</sup>Amazon Research

kemal.kurniawan@student.unimelb.edu.au

{lea.frermann, tcohn}@unimelb.edu.au

phschulz@amazon.com

## Abstract

This paper describes PTST, a source-free unsupervised domain adaptation technique for sequence tagging, and its application to the SemEval-2021 Task 10 on time expression recognition. PTST is an extension of the cross-lingual parsimonious parser transfer framework (Kurniawan et al., 2021), which uses high-probability predictions of the source model as a supervision signal in self-training. We extend the framework to a sequence prediction setting, and demonstrate its applicability to unsupervised domain adaptation. PTST achieves  $F_1$  score of 79.6% on the official test set, with the precision of 90.1%, the highest out of 14 submissions.<sup>1</sup>

## 1 Introduction

SemEval-2021 Task 10 presents source-free unsupervised domain adaptation (SFUDA) for semantic processing.<sup>2</sup> The goal of *unsupervised domain adaptation* is to transfer a model from a source domain to another, different domain — called target domain — using only unlabelled data in the target domain. *Source-free* unsupervised domain adaptation additionally assumes no access to source domain data: only the source model pre-trained on the source domain data is available. This situation may occur when the source domain data contains protected information that cannot be shared, or even if it can, requires signing a complex data use agreement. While there are numerous works on SFUDA outside NLP (Hou and Zheng, 2020; Kim et al., 2020; Liang et al., 2020; Yang et al., 2020), SFUDA research for NLP is severely lacking in spite of its importance in, for example, clinical

NLP (Laparra et al., 2020). There are two tasks involved in SemEval-2021 Task 10: negation detection and time expression recognition. We participate only in the latter.

Our approach is an extension of the parsimonious parser transfer framework (PPT; Kurniawan et al. (2021)). PPT allows cross-lingual transfer of dependency parsers in a source-free manner, requiring only unlabelled data in the target side. It leverages the output distribution of the source model to build a chart containing high probability trees for each sentence in the target data. We extend this work by (1) formulating PPT for chain structures and evaluating it on a semantic sequence tagging task; and (2) demonstrating its effectiveness in a domain adaptation setting. We call our method **Parsimonious Transfer for Sequence Tagging** (PTST).

We find PTST effective for improving the precision of the system in the target domain. It ranks 7th out of 14 submissions in the official leaderboard in terms of  $F_1$  score, but 1st in precision, with a gap of 3 points from the second best. Drawing on the model calibration literature, we provide a way to combat the problem of model overconfidence which is key to make PTST outperform a simple transfer of a source model to the target domain. However, we also find that PTST struggles in improving recall. In conclusion, our results suggest that PTST can be used for SFUDA, but further work is required to improve the precision-recall trade-off in the target domain.

## 2 Background

In the SemEval-2021 Task 10 time expression recognition task, the input is a single sentence, and the output is a sequence of tags indicating the time entity type of a word (if any). There are 32 time entities in total (e.g., Year, Hour-of-Day), and

\*Work done outside Amazon.

<sup>1</sup>Our code is available at <https://github.com/kmkurn/ptst-semeval2021>.

<sup>2</sup><https://competitions.codalab.org/competitions/26152>

○ ○ ○ ○ B-Day-of-Week B-Part-of-Day  
A woman was killed Thursday evening

Figure 1: An example input sentence and its output sequence of tags for the time expression recognition task.

the tags are coded in BIO format. Fig. 1 shows an example input sentence and its output. The task organisers provided a pre-trained source model for the task. This source model is RoBERTa-base (Liu et al., 2019) that is pre-trained on more than 25K time expressions in English clinical notes from Mayo Clinic in SemEval-2018 Task 6. The model is distributed online via HuggingFace Models,<sup>3</sup> which can be obtained with HuggingFace Transformers library.<sup>4</sup> The organisers also released trial data for the practice phase containing 99 annotated English articles from the news domain. The official test data released by the organisers in the evaluation phase contains 47 articles that are in a different domain from the source and development data.

The time expression recognition task is formalised as a sequence tagging task. The literature on sequence tagging in NLP is massive (Jiang et al., 2020; He and Choi, 2020; Rahimi et al., 2019; He et al., 2019; Xie et al., 2018; Clark et al., 2018, *inter alia*). One closely related task is named-entity recognition (NER) whose goal is to detect mentions of named entities such as a Person or Organisation in an input sentence. Lample et al. (2016) introduced a now widely adopted neural architecture for this task, where input word embeddings are encoded with a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) before they are passed through per-word softmax layers. In recent times, it is common to replace the LSTM with a Transformer (Vaswani et al., 2017). With the advancements of large pre-trained language models, the standard is to use a model such as BERT (Devlin et al., 2019) as the encoder and fine-tune the model on labelled data. The source model of the time expression recognition task provided by the organisers was also trained in this manner.

In the context of unsupervised domain adaptation, a popular approach is domain adversarial training. Introduced by Ganin et al. (2016), it leverages multi-task learning in which the model is optimised

not only on the main task objective but also a domain prediction objective. The learning signal for the latter is passed through a gradient reversal layer, ensuring that the learnt parameters are predictive for the main task, but general across domains. With pre-trained language models, Han and Eisenstein (2019) proposed to continue pre-training on the unlabelled target domain data, prior to finally fine-tuning on the labelled source domain data. Unfortunately, these approaches require access to the source domain data.

There is relatively little work on SFUDA in NLP, however, some works on source-free cross-lingual transfer exist. Wu et al. (2020) employ teacher-student learning for source-free cross-lingual NER. A teacher model trained on the source side predicts soft labels on the unlabelled target side data, and a student model is trained on those soft labels. Their method outperforms a simple direct transfer method where the source model is directly applied on the target side. More recently, a method for source-free cross-lingual transfer of dependency parsers was introduced by Kurniawan et al. (2021). The key idea is to build a chart of high probability trees based on arc marginal probabilities for each unlabelled sentence on the target side, and treat all those trees as a weak supervision signal for training. Their method outperforms direct transfer as well as a variety of recent cross-lingual transfer methods that are not source-free. That said, the effectiveness of their method on (a) semantic (sequence labelling) tasks and (b) in a domain adaptation setting is unexplored, which is what we aim to address in this work.

### 3 System Description

We first describe our sequence tagging model (Section 3.1), before we present parsimonious transfer for sequence tagging (PTST) in Section 3.2.

#### 3.1 Model

Our model is a linear-chain conditional random field (CRF) over tag sequences. It assigns a score  $s(\mathbf{x}, \mathbf{y})$  to a pair of input sentence  $\mathbf{x}$  and output tag sequence  $\mathbf{y}$ , which can be expressed as

$$s(\mathbf{x}, \mathbf{y}) = \sum_j \pi(\mathbf{x}, j, y_j) + \phi(y_j, y_{j+1}) \quad (1)$$

where  $\pi(\mathbf{x}, j, t)$  is the *emission* score of word  $x_j$  having tag  $t$  and  $\phi(t, t')$  is the *transition* score of having tag  $t$  followed by tag  $t'$ . The probability of

<sup>3</sup><https://huggingface.co/clulab/roberta-timex-semeval>

<sup>4</sup><https://github.com/huggingface/transformers>

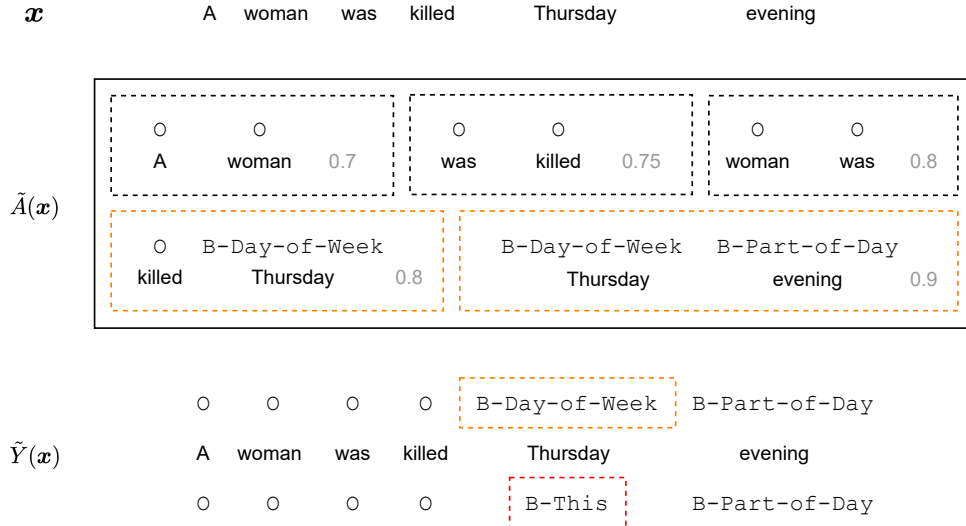


Figure 2: Illustration of our method. Given an unlabelled sentence  $\mathbf{x}$  from the target domain, we build the set of high probability tag pairs  $\tilde{A}(\mathbf{x})$  using the source model, which may contain correct tag pairs that do not occur in the predicted tag sequence (in orange). From these tag pairs, we build the chart  $\tilde{Y}(\mathbf{x})$  containing tag sequences that can be assembled from tag pairs in  $\tilde{A}(\mathbf{x})$ . The single predicted tag sequence from the source model (bottom) is also included in the chart, but it may contain an incorrect tag (in red) as it is noisy.

$\mathbf{y}$  given  $\mathbf{x}$  is then

$$P(\mathbf{y} | \mathbf{x}) \propto \exp s(\mathbf{x}, \mathbf{y}). \quad (2)$$

The emission score function  $\pi$  is parameterised by a neural network whose parameters are initialised with the source model. Specifically, the emission score function is the RoBERTa model provided by the task organisers. The transition score function  $\phi$  is a  $T \times T$  parameter matrix that is learned during training, where  $T$  is the number of tags. Note that with dynamic programming, we can efficiently compute quantities such as the marginal probabilities of tag pairs  $P((j, y_j, y_{j+1}) | \mathbf{x})$  or the partition function  $Z(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \exp s(\mathbf{x}, \mathbf{y})$  where  $\mathcal{Y}(\mathbf{x})$  is the set of all possible tag sequences for  $\mathbf{x}$ .

### 3.2 Unsupervised Adaptation

To perform unsupervised adaptation on the unlabelled target domain data, we extend PPT, our past work for source-free cross-lingual parsing (Kurniawan et al., 2021), to chain structures. Given a set of unlabelled sentences  $D$  in the target domain, we build a chart of high probability tag sequences  $\tilde{Y}(\mathbf{x})$  by leveraging the output distribution in the source model. The model then treats all sequences with sufficiently high predicted probability as possible tag sequences for  $\mathbf{x}$  for training. Concretely, it minimises the loss:

$$\ell(\theta) = - \sum_{\mathbf{x} \in D} \log \sum_{\mathbf{y} \in \tilde{Y}(\mathbf{x})} P_{\theta}(\mathbf{y} | \mathbf{x}) \quad (3)$$

where  $\theta$  denotes the target model parameters. The set  $\tilde{Y}(\mathbf{x})$  is defined formally as

$$\tilde{Y}(\mathbf{x}) = \{\mathbf{y} | \mathbf{y} \in \mathcal{Y}(\mathbf{x}) \wedge A(\mathbf{y}) \subseteq \tilde{A}(\mathbf{x})\} \quad (4)$$

where  $\mathcal{Y}(\mathbf{x})$  is the set of all possible tag sequences for  $\mathbf{x}$ ,  $A(\mathbf{y}) = \{(j, y_j, y_{j+1}) | 1 \leq j < |\mathbf{y}|\}$  is the set of consecutive tag pairs in tag sequence  $\mathbf{y}$ , and  $\tilde{A}(\mathbf{x}) = \bigcup_j \tilde{A}(\mathbf{x}, j)$  where  $\tilde{A}(\mathbf{x}, j)$  is the set of high probability consecutive tag pairs  $(j, t_j, t_{j+1})$  for words  $x_j$  and  $x_{j+1}$  (see Fig. 2 for illustration). Analogous to PPT, this set is constructed by adding tag pairs  $(t_j, t_{j+1})$  in order of decreasing marginal probability until their cumulative probability exceeds a threshold  $\sigma$ . The predicted tag sequence from the source model is also included in  $\tilde{Y}(\mathbf{x})$  so the chart is never empty.

Note that the method above is very similar to self-training where the predictions from the source model are used as supervision signal for training. In contrast to self-training, however, we build a chart of high probability predictions for each sample instead of just a single best prediction. We expect these predictions to be more useful than a single best prediction because it is more likely for the correct tag sequence to be in the chart than equal to the single best prediction. Even when this is not the case, we expect the partially correct tag sequences occur frequently enough in the chart so the model is still able to learn what the correct tag sequence is.

Team	$F_1$	P	R
BCLUFIGHT-1	<b>81.5</b>	84.7	78.5
Self-Adapter-1	81.1	87.3	75.7
BLCUFIGHT-2	81.0	83.4	78.7
Baseline-2	80.4	82.7	78.2
YNU-HPCC-2	80.3	81.7	79.1
Self-Adapter-2	79.7	83.9	76.0
<b>PTST-UoM-1 (ours)</b>	79.6	<b>90.1</b>	71.3
Boom-1	79.5	86.9	73.2
UArizona-1	79.5	78.6	80.4
UArizona-2	79.5	78.3	<b>80.7</b>
Baseline-1	79.4	84.9	74.6
KISNLP-1	79.3	81.0	77.7
KISNLP-2	78.1	79.8	76.4
YNU-HPCC-1	74.8	87.2	65.5

Table 1: TIMEX leaderboard on the test data.

In our preliminary experiments, we find that it is crucial to introduce temperature scaling to the emission scoring function in order to achieve good performance. Thus, for our main result, we define a new emission scoring function  $\pi'$  as

$$\pi'(\mathbf{x}, j, y_j) = \pi(\mathbf{x}, j, y_j) / \tau \quad (5)$$

where  $\tau$  is the temperature scale hyperparameter. We discuss and provide an analysis of this temperature scaling in Section 5.

## 4 Experimental Setup

We use the pre-trained source model and data provided for SemEval-2021 Task 10. We only use the model and data for the time expression recognition task (TIMEX hereinafter) as we only participate in that task. We use the practice data as the development set to tune the hyperparameters of our model with random search.<sup>5</sup> We set the threshold  $\sigma = 0.95$  following the setup of Kurniawan et al. (2021). We do not use any data sets other than those provided by the task organisers for TIMEX.

We train PTST on the unlabelled test data for 5 epochs. As described in Section 3.1, we initialise the neural network for the emission scoring with the source RoBERTa model provided by the task organisers. We enforce the BIO constraints by initialising the transition matrix  $\phi$  with  $-\infty$  for entries corresponding to illegal transitions, and zero otherwise.<sup>6</sup> We use the linear CRF implementation provided in the Torch-Struct library (Rush, 2020). To avoid out-of-memory error, we discard

<sup>5</sup>Best learning rate and  $\tau$  are  $9 \times 10^{-6}$  and 2.56 respectively.

<sup>6</sup>The constraints require an inside tag be always preceded by an inside or beginning tag of the same entity.

sentences longer than 30 tokens. Additionally, we find that it is useful to freeze the embedding and the first few layers of the RoBERTa encoder. Thus, in our main result, we freeze the embedding layer and the first 6 (out of 12) layers of the encoder. An analysis is provided in Section 5.

## 5 Results and Discussion

Table 1 shows the TIMEX leaderboard on the official test data in the evaluation phase.<sup>7</sup> Our model PTST is ranked 7th out of 14 submissions in terms of  $F_1$  score, below the baseline model submission by the task organisers which ranks 4th. This baseline model is the pre-trained source model fine-tuned on the labelled development data. Despite the relatively low  $F_1$  score, PTST achieves 90.1 % precision, which is the highest among all submissions, and markedly above the second highest precision of 87.3 % achieved by the second best performing model. Looking at recall, our model has the second lowest score of 71.3 %, which is fairly below the third lowest one of 73.2 %. This result suggests that our model is sacrificing recall in favour of precision, which may be a desirable property for downstream tasks where making the right prediction is more critical.

**Model overconfidence** As mentioned in Section 3.2, in our preliminary experiments, we find that the source model is extremely confident about its predictions, making the marginal probability distribution of tag pairs at any position  $j$  very sharp. This sharpness results in  $\hat{Y}(\mathbf{x})$  containing mostly just a single tag sequence, which is the predicted sequence from the source model, rendering the whole approach no different from simple self-training. To remedy this problem, we introduce temperature scaling in the emission score, which has been shown to be a simple but effective trick in model calibration (Geman and Geman, 1984; Guo et al., 2017). We define the new emission scoring function as shown in Eq. (5). Table 2 shows how the performance of PTST changes when  $\tau$  is varied. We see that as  $\tau$  increases, precision does too, but recall decreases, although in a relatively slower rate so the  $F_1$  score tends to increase as well. Also reported in Table 2 is the median number of tag sequences and the fraction of gold tag sequences

<sup>7</sup>Also available on <https://machine-learning-for-medical-language.github.io/source-free-domain-adaptation/leaderboard>.

$\tau$	$F_1$	P	R	$n$	$p$ (%)
1.0	$77.0 \pm 0.4$	$76.5 \pm 1.0$	$77.5 \pm 0.3$	1	54.1
1.5	$77.4 \pm 0.2$	$77.5 \pm 0.1$	$77.4 \pm 0.5$	1	56.9
2.0	$77.6 \pm 0.1$	$79.1 \pm 0.5$	$76.1 \pm 0.4$	1	64.0
2.5	$77.8 \pm 0.5$	$81.2 \pm 0.2$	$74.7 \pm 1.0$	$9.2 \times 10^{13}$	79.7
3.0	$26.7 \pm 7.9$	$96.3 \pm 2.9$	$15.7 \pm 5.3$	$2.2 \times 10^{17}$	86.9
SRC	77.1	77.5	76.8	—	—

Table 2: Model performance on the development data as  $\tau$  changes. SRC is the pre-trained source model directly applied on the development data.  $F_1$ , precision (P), and recall (R) scores are averages ( $\pm$  std) over 3 runs.  $n$  is the median number of high probability tag sequences in the chart  $\tilde{Y}(\mathbf{x})$ .  $p$  is the fraction of gold tag sequences contained in the chart.

	$F_1$	P	R
SRC	77.1	77.5	76.8
No freezing	$77.8 \pm 0.5$	$81.2 \pm 0.2$	$74.7 \pm 1.0$
Freeze emb	$77.7 \pm 0.2$	$81.0 \pm 0.4$	$74.7 \pm 0.6$
Freeze emb + 6 layers	$78.2 \pm 0.2$	$80.3 \pm 0.3$	$76.2 \pm 0.0$
Freeze emb + 12 (all) layers	$77.2 \pm 0.0$	$77.7 \pm 0.0$	$76.7 \pm 0.0$

Table 3: Model performance on the development data when the RoBERTa embedding and encoder layers are frozen during training. SRC is the pre-trained source model directly applied on the development data. Scores are averages ( $\pm$  std) over 3 runs.

contained in the chart. The two quantities grow as  $\tau$  does, which indicates that increasing  $\tau$  indeed allows the chart to contain more tag sequences, and thus increasing the coverage of correct tag sequences in the chart. However, when  $\tau$  is too large ( $\tau = 3.0$ ), the model breaks down, presumably because  $\tilde{Y}(\mathbf{x})$  contains too many noisy tag sequences to be useful.

The decline in recall might be explained by the nature of the task, where in a single sentence most of the words are not time entities. When  $\tau$  grows, the number of high probability tag sequences in  $\tilde{Y}(\mathbf{x})$  does too. In the majority of these tag sequences, a word in  $\mathbf{x}$  is likely to be tagged as a non-entity because time entities are naturally rare. Since tag sequences are treated uniformly (i.e. no tag sequence weighs more than the others), this provides a strong signal for the model that the word is a non-entity. Therefore, the model’s capability of recognising entities is reduced. Conversely, a similar argument may explain the rise in precision. When the model predicts a word as an entity, it is likely that in the majority of tag sequences in  $\tilde{Y}(\mathbf{x})$ , the word is tagged as the same entity, providing a strong signal that the word is indeed that entity. In other words, if the model predicts an entity, the model is very confident about it. When confidence is high, it is more likely that the prediction is correct, thus resulting in higher precision.

**Freezing layers** We also find that it is helpful to freeze the embedding layer and the first few layers of the RoBERTa model’s encoder during training, presumably because they encode low-level linguistic information that is invariant across domains. Table 3 reports how the model performance changes with varying numbers of layers frozen ( $\tau$  is fixed to 2.5). We observe that freezing the embedding and first several encoder layers gives a small boost to performance, with best performance reached with 6 frozen layers (the setting adopted in the model reported in the main results).

**Error analysis** To better understand the errors of PTST, we present the confusion matrix of the model on the test data in Fig. 3. We see that the majority of the errors arise from the model not recognising actual time entities, consistent with the relatively low recall. The model has serious difficulties in recognising `Season-Of-Year`, for example, in fragments like:

*The increase in food aid beneficiaries is partly attributed to **Meher** harvest loss [...]* (1)

*The increase, which follows a **seasonal** trend, is seen in all regions except Tigray.* (2)

The model also seems to struggle with recognising `Between` and `This`. Example sentence fragments where the model wrongly predicts a

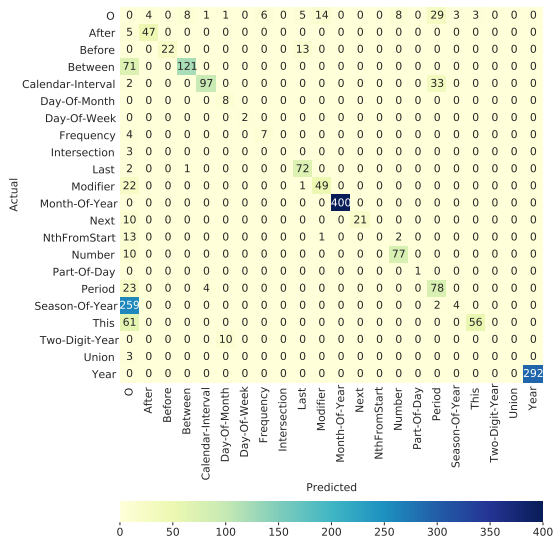


Figure 3: Confusion matrix on the test data.

non-entity are:

*In Gambella region, **between** 1 and 12 April, 3,346 South Sudanese refugees arrived [...]* (3)

where the model fails to recognise `Between`, and *[...] to prevent a potential national outbreak of AWD during **this** rainy season* (4)

where the model fails to recognise `This`. Although the model has good precision, we still see that it misclassifies non-entities and `Calendar-Interval` as `Period` relatively often. For example, in the fragment

*[...] the malnutrition situation is expected to aggravate in the coming **months*** (5)

the word *months* is a `Calendar-Interval` but the model predicts it as `Period`. Another example, the model predicts the word *period* in the sentence fragment

*During the reporting **period**, an estimated 1,000 south Sudanese arrived [...]* (6)

as `Period`, while the word is actually not a time entity.

## 6 Conclusions

We present PTST, our submission to the time expression recognition task of SemEval-2021 Task 10. We describe our sequence tagging model as a CRF over chain structures, parameterised by a neural network. Our domain adaptation approach leverages the output distribution of the source model to build a chart of high probability tag sequences for every sentence in the unlabelled target domain data.

PTST ranks 7th in terms of  $F_1$  score in the official leaderboard, but achieves the highest precision out of 14 submissions. We provide analyses on the importance of temperature scaling to mitigate model overconfidence and the patterns of errors.

## Acknowledgments

A graduate research scholarship is provided by Melbourne School of Engineering to Kemal Kurniawan.

## References

- Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc V. Le. 2018. [Semi-supervised sequence modeling with cross-view training](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. [Domain-adversarial training of neural networks](#). *The Journal of Machine Learning Research*, 17(1):2096–2030.
- S. Geman and D. Geman. 1984. [Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.
- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. [On calibration of modern neural networks](#). In *International Conference on Machine Learning*, pages 1321–1330.
- Xiaochuang Han and Jacob Eisenstein. 2019. [Unsupervised domain adaptation of contextualized embeddings for sequence labeling](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4238–4248.
- Han He and Jinho Choi. 2020. [Establishing strong baselines for the new decade: Sequence tagging, syntactic and semantic parsing with BERT](#). In *The Thirty-Third International Flairs Conference*.
- Junxian He, Zhisong Zhang, Taylor Berg-Kirkpatrick, and Graham Neubig. 2019. [Cross-lingual syntactic](#)

- transfer through unsupervised adaptation of invertible projections. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3211–3223.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Yunzhong Hou and Liang Zheng. 2020. Source free domain adaptation with image translation. *arXiv:2008.07514 [cs, eess]*.
- Zhengbao Jiang, Wei Xu, Jun Araki, and Graham Neubig. 2020. Generalizing natural language analysis through span-relation representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2120–2133.
- Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyoungseob Park, and Priyadarshini Panda. 2020. Domain adaptation without source data. *arXiv:2007.01524 [cs, eess]*.
- Kemal Kurniawan, Lea Frermann, Philip Schulz, and Trevor Cohn. 2021. PPT: Parsimonious parser transfer for unsupervised cross-lingual adaptation. *arXiv:2101.11216 [cs]*.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.
- Egoitz Laparra, Steven Bethard, and Timothy A. Miller. 2020. Rethinking domain adaptation for machine learning over clinical language. *JAMIA Open*, 3(2):146–150.
- Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692 [cs]*.
- Afshin Rahimi, Yuan Li, and Trevor Cohn. 2019. Massively multilingual transfer for NER. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 151–164.
- Alexander Rush. 2020. Torch-Struct: Deep structured prediction library. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 335–342.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.
- Qianhui Wu, Zijia Lin, Börje Karlsson, Jian-Guang LOU, and Biqing Huang. 2020. Single-/multi-source cross-lingual NER via teacher-student learning on unlabeled data in target language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6505–6514.
- Jiateng Xie, Zhilin Yang, Graham Neubig, Noah A. Smith, and Jaime Carbonell. 2018. Neural cross-lingual named entity recognition with minimal resources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 369–379.
- Shiqi Yang, Yaxing Wang, Joost van de Weijer, Luis Herranz, and Shangling Jui. 2020. Unsupervised domain adaptation without source data by casting a BAIT. *arXiv:2010.12427 [cs]*.

# Self-Adapter at SemEval-2021 Task 10: Entropy-based Pseudo-Labeler for Source-free Domain Adaptation

Sangwon Yoon<sup>1,2</sup>, Yanghoon Kim<sup>1,3</sup> and Kyomin Jung<sup>1,3</sup>

<sup>1</sup>Seoul National University, Seoul, Korea

<sup>2</sup>School of Law, Seoul National University, Seoul Korea

<sup>2</sup>Department of Electrical and Computer engineering, Seoul National University, Seoul Korea  
{sangwon38383, ad26kr, kjung}@snu.ac.kr

## Abstract

Source-free domain adaptation is an emerging line of work in deep learning research since it is closely related to the real-world environment. We study the domain adaption in the sequence labeling problem where the model trained on the source domain data is given. We propose two methods: Self-Adapter and Selective Classifier Training. Self-Adapter is a training method that uses sentence-level pseudo-labels filtered by the self-entropy threshold to provide supervision to the whole model. Selective Classifier Training uses token-level pseudo-labels and supervises only the classification layer of the model. The proposed methods are evaluated on data provided by SemEval-2021 task 10 and Self-Adapter achieves 2<sup>nd</sup> rank performance.

## 1 Introduction

Domain adaptation (DA) is the task of applying an algorithm trained on a source domain data to a different target domain data with limited/undefined labels. DA has gotten significant attention as an alternative of fine-tuning approach (Ganin and Lempitsky, 2015; Saito et al., 2018; Tzeng et al., 2017), especially in situations rich supervision is not possible (Morero et al., 2018). DA is an important way of overcoming the data shortage of deep learning since it enables the utilization of knowledge from other labeled data.

Source-free DA is then proposed to cope with such data sharing in the general setting of DA, the data distribution in the source domain and the target domain are related but different (Storkey and Sugiyama, 2007), and annotated samples from the source domain are available during the training process. However, many of the data resources are not allowed to be shared in real-life environments as there are increasing concerns for privacy issues. For example, Twitter has a regulation that prevents

sharing tweet text. The policy is even more rigorous in the financial/clinical domain under the privacy protection issue.

Unlike conventional DA, one can not get access to the source domain data in source-free DA but is provided a model trained on the source domain data. About source-free DA in computer vision, several approaches have been proposed; (Sahoo et al., 2020) assumes the target domain data is a transformation from the source domain data along natural axes such as brightness and contrast; (Kundu et al., 2020) proposes universal DA that is trained via two-stage learning of procurement and deployment; (Kim et al., 2020) progressively updates the target model with pseudo-labels which are selected under self-entropy criterion.

As for natural language processing (NLP), the application of source-free DA is slightly more complicated since sentences are usually considered as having discrete representations. In this context, SemEval-2021 task 10 has proposed a challenge that is related to source-free domain adaptation for semantic processing.

In this paper, we propose **Self-Adapter** for the *time expression recognition* sub-task in SemEval-2021 task 10. Following (Kim et al., 2020), we employ pseudo-labels from the target domain to further supervise the model trained on the source domain data, while the entropy-based evaluation of reliable pseudo-labels is adopted in consideration of the discrete text data. In addition, we adopt *Sloughing* trick to prevent over-fitting.

To demonstrate the efficacy of the Self-Adapter, we evaluate the proposed method on the dataset by (Laparra et al., 2018). We also compare the proposed method with several variations and another method we come up with, named **Selective Classifier Training (SCT)**. In the end, the Self-Adapter has achieved 2<sup>nd</sup> rank in the official evaluation period getting 0.811  $F_1$  which is 1.7 percentage



points higher than the RoBERTa-based sequence tagging model pre-trained only on source data.

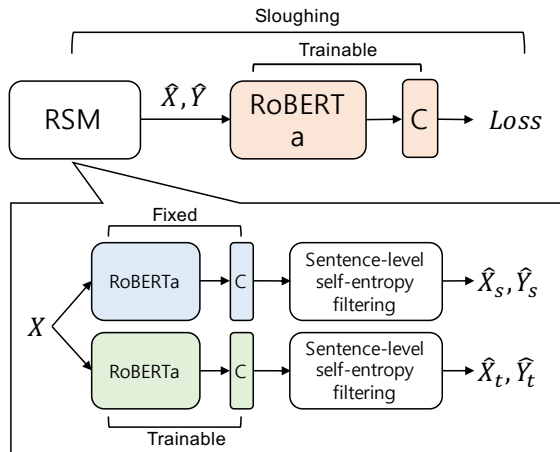


Figure 1: Training pipeline of Self-Adapter. At the beginning of every stage of training, RSM is initialized with ‘reliable’ samples generated by both fixed and trainable models. A trainable model is supervised using samples - *source-oriented pseudo-labels* and *target-oriented pseudo-labels* - stored in RSM.

## 2 Systems Description

Our proposed methods have three operations in common: (1) generating pseudo-labels, (2) filtering out pairs of reliable samples and pseudo-labels based on models’ self-confidence, and (3) doing supervised learning using the pseudo-labels. We concentrate on sorting out ‘reliable’ pseudo-labels since training with incorrect labels harms the performance of the model.

Self-entropy is usually treated as an indicator of self-confidence (Zou et al., 2018; Saporta et al., 2020). We adopt normalized self-entropy as the evaluation metric for pseudo-labels:

$$H(x_t) = -\frac{1}{\log N_c} \sum l(x_t) \log(l(x_t)) \quad (1)$$

where  $x_t$  denotes each token that makes up a sentence  $X \in \mathbf{X}$ .  $l(x_t)$  denotes the predicted probability of the predicted label by the classifier, and  $N_c$  refers to the total number of labels.

Specifically, we propose two adaptation methods to efficiently fit the model trained on a source domain to a target domain: Self-Adapter and SCT.

### 2.1 Method 1: Self-Adapter

We propose Self-Adapter which is a self-learning method under the supervision of reliable sample

memory (RSM). RSM is a set of data with pseudo-labels that consists of two parts, *source-oriented pseudo-labels* and *target-oriented pseudo-labels*, and each of them represents the knowledge learned from the source domain and new features to learn from the target domain. We further apply a trick called ‘*Sloughing*’ which helps prevent over-fitting. The overall workflow of Self-Adapter is shown in Figure 1.

#### 2.1.1 Reliable Sample Memory

RSM is the pairs of input sentences and the corresponding pseudo-labels obtained from a Siamese-like network structure. Two RoBERTa-based (Liu et al., 2019) classifiers are initialized with a RoBERTa-based sequence tagging model fine-tuned only on source train data, which is given as a baseline model in the task. One of the branch maintains fixed weight parameters while another is fine-tuned during training.

Both branches of the network take a target domain sentence  $X$  as an input and output a set of probabilities for labels each token should be assigned to. We utilize the self-entropy as the evaluation metric for the self-confidence of each token. If the self-confidence of each token is smaller than the predefined threshold, the pair of input sentences with the pseudo-labels generated by the model is kept as a part of RSM.

The fixed part of the network consistently outputs the same pairs  $(\hat{X}_s, \hat{Y}_s)$  which are called *source-oriented pseudo-labels*. The trainable part of the network outputs different pairs  $(\hat{X}_t, \hat{Y}_t)$  called *target-oriented pseudo-labels* after each update and both are stored in the RSM. All sentence-label pairs in RSM, both *source-oriented pseudo-labels* and *target-oriented pseudo-labels*, are used to train the trainable part of the network in a supervised manner. We call the cycle in which RSM are updated as a stage and each stage is composed of several epochs.

#### 2.1.2 Sloughing trick

After sufficient update of RSM, we generate pseudo-labels with RSM and do another self-entropy filtering to gain new reliable samples. Subsequently, we re-initialize the trainable part of the network with the parameter of the baseline and train it under the supervision of the new reliable samples. We call this procedure *Sloughing*. Since many of the reliable samples in each RSM update overlaps, over-fitting tends to happen over time.

The *Sloughing* then efficiently prevents over-fitting by newly initializing a model which is not fitted to test data yet.

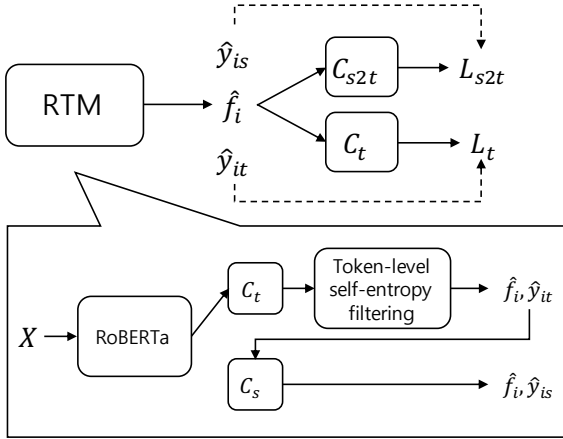


Figure 2: Training pipeline of Selective Classifier Training. RTM is updated at the start of each step of training. In multi-branch network whose two branches  $C_{s2t}$  and  $C_t$  share a fixed RoBERTa-based feature extractor, loss of  $C_{s2t}$  branch is calculated by supervision with *source-oriented token-wise pseudo-labels* and loss of  $C_t$  branch is calculated by supervision with *target-oriented token-wise pseudo-labels*.

## 2.2 Method 2: Selective Classifier Training

Selective Classifier Training is a training method that consists of a RoBERTa-based feature extractor and multi-branch classifiers. The feature extractor and classifiers are initialized with the RoBERTa-based sequence tagging model fine-tuned only on source train data, given as baseline model in the development phase. In SCT, only the classifiers are updated under the supervision of Reliable Token Memory (RTM).

### 2.2.1 Reliable Token Memory

RTM is the pairs of tokens and their pseudo-labels obtained from a network with two separate branches. Both the branches share the fixed feature extractor which is the same with the feature extractor of the SCT training pipeline. Two classifiers, a trainable classifier  $C_t$  and a fixed classifier  $C_s$ , make predictions on contextual embedding passed from the feature extractor.

To update RTM, we first get contextual embeddings for all tokens in target domain sentences by putting in all sentences as input of shared feature extractor and we get prediction on each token embeddings. Token embeddings  $\hat{f}_i$  whose normalized self-entropy predicted by  $C_t$  are lower

than the threshold  $\theta$  are called *reliable token-wise samples*. The pseudo-labels of reliable token-wise samples predicted by  $C_s$  are called *source-oriented token-wise pseudo-labels*. The pseudo-labels of reliable token-wise samples predicted by  $C_t$  are called *target-oriented token-wise pseudo-labels*. The pairs of *reliable token-wise samples* and their *source-oriented token-wise pseudo-labels*, and the pairs of *reliable token-wise samples* and their *target-oriented token-wise pseudo-labels* consists RTM.

### 2.2.2 Multi-branch network

With RTM, we train a multi-branch network in which each branch shares a fixed feature extractor. They divide into two classifiers  $C_{s2t}$  and  $C_t$ . Loss of  $C_{s2t}$  branch is calculated by supervision with *source-oriented token-wise pseudo-labels* and loss of  $C_t$  branch is calculated by supervision with *target-oriented token-wise pseudo-labels*. RTM updates at the start of each step of training.

The loss function is formulated as

$$L_{total} = (1 - \alpha)L_{s2t} + \alpha L_t \quad (2)$$

where  $L_{s2t}$  and  $L_t$  indicates loss function of  $C_{s2t}$  branch and  $C_t$  respectively.  $\alpha$  is a weight between two branches. We gradually increase  $\alpha$  from 0 to 1 to deal with high instability in the early stages of learning, in the same way as (Kim et al., 2020). In the test phase, we use the classification probability of the  $C_t$  branch.

## 3 Experiments

We evaluate our two models: Self-Adapter, SCT and their variations. The baseline on the development data is a RoBERTa-based sequence tagging model pre-trained on only the source data: de-identified clinical notes from the Mayo Clinic, called Source-Trained. Also, there is another baseline Dev-Tuned on the test data which is the source pre-trained model (i.e., Source-Trained) fine-tuned on the labeled development data. The development data is the annotated news portion of the SemEval-2018 Task 6 data. Test data is a set of annotated documents extracted from food security warning systems. development data consists of 1580 sentences and test data consists of 3911 sentences. The total number of labels is 65, where label 0 indicates non-time entity, and label 1-64 indicates different types of time entities.

Method	$F_1$	<i>Precision</i>	<i>Recall</i>
SCT	0.784	0.814	0.756
SA	0.808	0.819	0.797
SA+ <i>Sloughing</i> *	<b>0.812</b>	0.822	0.801
SA-filtering	0.771	0.774	0.768
Source-Trained	0.771	0.768	0.775

Table 1:  $F_1$ , *Precision*, *Recall* on development data. The model submitted to the competition is marked with \*. SA indicates Self-Adapter and SA+*Sloughing* is a system where *Sloughing* is applied on a model trained with Self-Adapter. SA-filtering is a system whose training pipeline is the same as Self-Adapter except that confidence filtering is not done. Source-Trained is a RoBERTa-based sequence tagging model pre-trained on only the source data: de-identified clinical notes from the Mayo Clinic, given as baseline model in the development phase of the competition.

### 3.1 Experimental setup

For all of our models, we set normalized self-entropy threshold  $\theta = 0.1$  except when applying *Sloughing* trick, on which  $\theta = 0.01$ . We train Self-Adapter for 3 stages. Each iteration consists of 4 epochs with batch size 1 (sentence-level) and the learning rate is fixed as  $5e-5$ . In Self-Adapter, pseudo-labels are updated at every stage. In Self-Adapter combined with *Sloughing*, we apply *Sloughing* for 3 times, 4 epochs training with batch size 1 (sentence-level) is done every time. The learning rate is fixed as  $5e-5$ . In SCT, pseudo-labels are updated every epoch. We train 2 epochs with batch size 4 (token-level) and the learning rate is scheduled with inverse decay scheduler same as (Kim et al., 2020), with initial learning rate  $5e-5$ . We use Adam optimizer in all models.

### 3.2 Experimental results and analysis

Table 1 and Table 2 shows the performance of the proposed methods on development and test data respectively. Each method is evaluated with *Precision*, *Recall*, and  $F_1$ . *Precision* is the ratio of correctly predicted positive observations to the total predicted positive observations. *Recall* is the ratio of correctly predicted positive observations to all observations in the actual class.  $F_1$  is the weighted average of *Precision* and *Recall*. Our major concern is  $F_1$ , which is the most preferred indicator of accuracy in text classification tasks.

On both data, Self-Adapter combined with *Sloughing* performs the best in  $F_1$  and Self-Adapter performs the second-best. SCT does not provide

Method	$F_1$	<i>Precision</i>	<i>Recall</i>
SA	0.81	0.874	0.754
SA+ <i>Sloughing</i> *	<b>0.811</b>	0.873	0.757
Source-Trained	0.794	0.849	0.746
Dev-Tuned	0.804	0.827	0.782

Table 2:  $F_1$ , *Precision*, *Recall* on the test data. The model submitted to the competition is marked with \*. SA indicates Self-Adapter and SA+*Sloughing* is a system where *Sloughing* is applied on a model trained with Self-Adapter. Source-Trained is a RoBERTa-based sequence tagging model pre-trained on only the source data: de-identified clinical notes from the Mayo Clinic and Dev-Tuned is a the source pre-trained model (i.e., Source-Trained) fine-tuned on the labeled development data.

significant improvement of  $F_1$  compared to Self-Adapter. Self-Adapter without confidence filtering performs almost the same as Source-Trained on every evaluation metric.

#### 3.2.1 Impact of confidence filtering

Our confidence filtering proves to be effective in dealing with the uncertainty of pseudo-labels. Self-Adapter, whose core is confidence filtering, increases 3.7, 1.6 percentage points of  $F_1$  on development data and test data for each. The system whose training pipeline is the same as Self-Adapter except that confidence filtering is not done performs almost the same as the Source-Trained.

#### 3.2.2 Necessity of training feature extractor

Well-trained BERT embeddings contain both syntactic (Hewitt and Manning, 2019) and semantic (Coenen et al., 2019) information of words. However, this is only when the model is fine-tuned with data from the domain same as the target domain. It is well known that embedding models trained on different domains poorly capture the domain-specific vocabularies and word semantics due to domain shift. (Sarma et al., 2018)

Since RoBERTa is a BERT-based language model, the same issue arises on RoBERTa used in this task. Thus if the feature extractor used for embedding words is fixed during training, the embeddings obtained do not provide sufficient information to the classifier, resulting in a limitation to improving performance. This is also shown through experimental results in which Self-Adapter outperforms SCT.

### 3.2.3 Inefficiency of *Sloughing*

In Self-Adapter, the model learns from almost all sentences in development data and test data. Only 333 sentences out of 1580 and 917 sentences out of 3911 were filtered in development data and test data for each despite the high threshold we set ( $\theta = 0.1$ ). It affects the magnitude of the effect of the *Sloughing* in our method. *Sloughing* improves performance on both development and test data, but not enough to be taken as meaningful. 0.04 percentage points of  $F_1$  on development data and 0.1 percentage points of  $F_1$  on test data increase by application of *Sloughing*.

Somewhat discouraging effect of *Sloughing* is due to the setting of our task, in which training is done with almost all samples in test data, despite confidence filtering. We expect *Sloughing* to be more effective in the setting where the bigger proportion of samples are filtered and thus the ability for generalization on unseen data is more important. However, verification of these hypotheses will be carried out as a follow-up study.

## 4 Conclusion

In this paper, we propose novel training methods Self-Adapter and Selective Classifier Training that improve model performance on the target domain only by leveraging the RoBERTa-based model pre-trained on source data. Both models rely on self-learning with highly credible pseudo-labels that are filtered based on self-entropy, differ only in the range of trainable parts. Also, we propose *Sloughing* trick to prevent over-confidence of the model by softening the network output. Our work is highly applicable in the real world since we have achieved remarkable improvement in performance using only a few test data which is not annotated at all, without any manual supervision.

## Acknowledgments

This work was supported by the Technology Innovation Program (10073144, Developing machine intelligence based conversation system that detects situations and responds to human emotions) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea)

## References

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Viégas, and Martin Watten-

berg. 2019. Visualizing and Measuring the Geometry of BERT. In *proceedings of the 33rd Conference on Neural Information Processing Systems*.

Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

John Hewitt and D Christopher Manning. 2019. A Structural Probe for Finding Syntax in Word Representations. In *proceedings of Association for Computational Linguistics*.

Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyoungeob Park, and Priyadarshini Panda. 2020. Domain Adaptation without Source Data. *arXiv preprint arXiv:2007.01524*.

Jogendra Nath Kundu, Naveen Venkat, R Venkatesh Babu, et al. 2020. Universal Source-Free Domain Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4544–4553.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. Semeval 2018 task 6: Parsing Time Normalizations. In *proceedings of the 12th International Workshop on Semantic Evaluation*, pages 88–96.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. 2018. Minimal-Entropy Correlation Alignment for Unsupervised Deep Domain Adaptation. In *proceedings of International Conference on Learning Representations*.

Roshni Sahoo, Divya Shanmugam, and John Guttag. 2020. Unsupervised Domain Adaptation in the Absence of Source Data. *arXiv preprint arXiv:2007.10233*.

Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Maximum Classifier Discrepancy for Unsupervised Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732.

Antoine Saporta, Tuan-Hung Vu, Matthieu Cord, and Patrick Pérez. 2020. Esl: Entropy-guided Self-supervised Learning for Domain Adaptation in Semantic Segmentation. In *proceedings of Computer Vision and Pattern Recognition*.

Prathusha Sarma, K, Yingyu Liang, and William Sethares, A. 2018. Domain Adapted Word Embeddings for Improved Sentiment Classification. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*.

Amos Storkey and Masashi Sugiyama. 2007. Mixture Regression for Covariate Shift. In *proceedings of Neural Information Processing Systems*, pages 1337–1344.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, page 7167–7176.

Yang Zou, Zhiding Yu, Xiaofeng Liu, Vijayakumar Bhagavatula, and Jinsong Wang. 2018. Confidence Regularized Self-Training. In *proceedings of Computer Vision and Pattern Recognition*.

# The University of Arizona at SemEval-2021 Task 10: Applying Self-training, Active Learning and Data Augmentation to Source-free Domain Adaptation

**Xin Su**

School of Information  
University of Arizona  
xinsu@email.arizona.edu

**Yiyun Zhao**

Department of Linguistics  
University of Arizona  
yiyunzhao@email.arizona.edu

**Steven Bethard**

School of Information  
University of Arizona  
bethard@arizona.edu

## Abstract

This paper describes our systems for negation detection and time expression recognition in SemEval 2021 Task 10, Source-Free Domain Adaptation for Semantic Processing. We show that self-training, active learning and data augmentation techniques can improve the generalization ability of the model on the unlabeled target domain data without accessing source domain data. We also perform detailed ablation studies and error analyses for our time expression recognition systems to identify the source of the performance improvement and give constructive feedback on the temporal normalization annotation guidelines.

## 1 Introduction

Unsupervised Domain Adaptation (UDA) is a task that generalizes knowledge acquired from a model trained on labeled data in one domain (source domain) to unlabeled data in a different domain (target domain). Conventional UDA algorithms usually require access to both source-domain and target-domain data (Ganin et al., 2016; Glorot et al., 2011; Chen et al., 2012; Louizos et al., 2016). However, sharing source-domain data is often not practical for clinical texts due to their highly sensitive personal information and complex data use agreement procedures (Laparra et al., 2020). To overcome this difficulty, Laparra et al. (2020) propose a new task of source free domain adaptation (SFDA) where only models trained on source-domain data are shared, which allows the possibility of using the information from the source-domain while reducing private information leakage. The biggest challenge of this task is to transfer task-related information embedded in the trained models.

Our team participated in both subtasks of SemEval 2021 Task 10 (Laparra et al., 2021), Source Free Domain Adaptation for Semantic Processing: negation detection and time expression recognition.

For both tasks, participants were given a RoBERTa model (Liu et al., 2019) fine-tuned on the source-domain, and asked to make predictions in the target-domain.

The goal of the negation detection task is to predict whether an event in a sentence is negated by its context. This is a binary sentence classification task. For example, given the event *diarrhea* and the sentence *Has no diarrhea and no new lumps or masses*, the goal is to predict that *diarrhea* is negated by its context.

The goal of time expression recognition sub-task (Laparra et al., 2018) is to recognize time expressions in the target domain. This is a named entity recognition (NER) task. The number of entity types (inside–outside–beginning format) is 65. Entity types in this task are formally defined time entity types from the Semantically Compositional Annotation of Time Expressions (SCATE) (Bethard and Parker, 2016) annotation schema. For example, in *2021-02-19*, *2021* will be labeled as Year, *02* will be labeled as Month-Of-Year and *19* will be labeled as Day-Of-Month.

We investigate self-training, active learning, and data augmentation techniques on negation detection and time expression recognition under the SFDA setting. Our contributions are:

1. We demonstrate that simple self-training over a small portion of the target domain data can effectively improve the performance of the negation detection model.
2. We demonstrate that active learning with data augmentation can significantly improve time expression recognition performance when selected examples are accurately annotated.
3. We perform ablation studies for the time expression recognition systems to analyze where the performance improvement comes from.
4. We analyze our annotation errors for the time recognition task and give constructive feed-

back on the annotation guideline and schema.

## 2 System Description

The source-domain models for both subtasks are RoBERTa-base models with linear classification output layers, implemented via the Huggingface Transformers library (Wolf et al., 2020), using RobertaForSequenceClassification for negation, and RobertaForTokenClassification for time.

The input to the models is a sequence tokenized by Byte-Pair Encoding (BPE). Following the conventions of the RoBERTa model input format, two special tokens  $\langle s \rangle$  and  $\langle /s \rangle$  are inserted at the beginning and end of the sequence, respectively. In the negation detection task, targeted events are denoted with two special tokens  $\langle e \rangle$  and  $\langle /e \rangle$  that are inserted before and after the event. For example, the sentence *Has no diarrhea and no new lumps or masses* with event *diarrhea* will be converted to  $\langle s \rangle$ *Has no*  $\langle e \rangle$ *diarrhea*  $\langle /e \rangle$  *and no new lumps or masses.*  $\langle /s \rangle$ . The model output for negation detection is whether the target event is negated and the model output for time expression detection is the labels for each input tokens.

### 2.1 Negation Detection System

We employ a simple self-training (Yarowsky, 1995) approach that fine-tunes the model with its own predictions on the unlabeled dataset. We start with the pre-trained source-domain model,  $M$ . Then, for each self-training iteration:

1. We initialize an empty training set,  $L$ .
2. We use  $M$  to label the target domain data.
3. If an instance is labeled with a probability above a threshold  $\tau$ , we add it to  $L$  with the predicted label as its pseudo label.
4. We fine-tune  $M$  on  $L$ .

When the source-domain model predictions are the same for two consecutive iterations or the number of iterations of self-training is greater than the predefined maximum number, self-training stops. Note that the training set  $L$  is reinitialized at each iteration, and the model is iteratively fine-tuned.

### 2.2 Time Expression Detection System

Our approach combines active-learning (Cohn et al., 1996) and data-augmentation (Simard et al., 2003). We start with the pre-trained source-domain model,  $M_0$ , a copy of the pre-trained source-domain model,  $M$ , and initialize an empty training set  $L$ . Then, for each iteration:

1. We select the  $k$  instances where  $M$  is most uncertain, manually annotate them, and add them to  $L$ . (Details in section 2.2.1.)
  2. We augment each manually annotated instance with  $n$  new examples and add them to  $L$ . (Details in section 2.2.2.)
  3. We re-initialize  $M$  to  $M_0$  and fine-tune on  $L$ .
- We repeat this process  $i$  times. Note that the training set  $L$  is built cumulatively, and  $M$  is reinitialized on each iteration.

#### 2.2.1 Active Learning

We use active learning methods to manually label the most uncertain examples of the model in the target domain. We believe that it is not practical to manually label the entire target domain dataset during the test phase. This requires sufficient expertise and time from annotators (we show later that it is very difficult to understand annotation guidelines in a short time). Otherwise, low-quality annotations will hurt the performance of the model. In each iteration, we select the top  $k$  target domain sentences with the highest uncertainty scores to manually annotate. We define the uncertainty score of an example as the sum of the model’s prediction’s entropy for each token within the sentence. Manual annotation follows the SCATE annotation guidelines released by the organizers.

The annotators were the first two authors of this paper, a Linguistic PhD student and a Information PhD student. During the annotation process, we first individually annotated examples and then resolved annotation differences through discussion. Our first exposure to the SCATE annotation schema was approximately 10 days before the start of the test phase, when we began reading the guidelines and posting questions on the Google forum. We used gold annotations from the development set (on the news domain) to simulate the annotation process during the practice phase. We believe this is similar to most real-world SFDA situations, where the person applying the model on the target domain is unfamiliar with the annotation guidelines and has limited time to learn them.

#### 2.2.2 Data Augmentation

Inspired by Miao et al. (2020), we applied data augmentation to increase the size of our training set beyond what can be achieved by manual annotation, and to improve the generalization of the model. For each time entity that we manually annotated, we automatically generated new training examples by

Task	Type	Domain	#	# of labeled	Open to participants
Negation	Train	-	-	all	✗
Negation	Dev	Clinical	8431 sentences	5545 sentences	✓
Negation	Test	Clinical	622703 sentences	9580 sentences	✓
Time	Train	-	-	all	✗
Time	Dev	news	99 documents	all	✓
Time	Test	food security	48 documents	17 documents	✓

Table 1: Data summary for negation detection and time expression recognition tasks

substituting original time entities with entities of the same type randomly sampled from a predefined entity candidates pool. We generate up to  $n$  new sequences (the size of the pool may be less than  $n$  for some entities). For example, if we manually annotate the three time entities from *2021-02-19* (*2021*: Year, *02*: Month-Of-Year, *19*: Day-Of-Month) in a sentence, after data augmentation, it can generate up to  $n \times 3$  additional sequences with different years, months and days (e.g., *2020-10-05*). The entity candidates pool is created based on the time entities in the development set and the annotation guideline. We filtered out entities that do not appear in the unlabeled test set data during the testing phase. See appendix A.1 for the final pool.

### 3 Data

All data used is in English. Both subtasks had training, development and test data, each representing different domains. As participants, we did not have access to the training set. The training sets are used by organizers to fine-tune the pre-trained RoBERTa-base models to obtain the source-domain models. We used the source-domain models and development sets to develop source-free domain adaptation systems during the practice phase, and tested our systems during test phase. We summarize the data in table 1.

## 4 Experiments

The organizers provided two baseline models for each task: the source-domain model, and the source-domain model fine-tuned on the development set. The official evaluation metric is the F1 score. Precision and recall scores are also reported.

### 4.1 Negation Detection

In the testing phase, we first fine-tuned the source-domain model on the labeled development set. Although the domains of the development set and

Strategy	F1	Precision	Recall
<i>Test Phase</i>			
SD	0.660	0.917	0.516
SD + FT	0.730	0.908	0.611
SD + FT + ST	<b>0.767</b>	0.880	0.680

Table 2: The performance of the negation detection systems during test phase. SD is the source-domain model. FT is fine-tuning on the development set. ST is self-training on test set.

the test set are different, they are both clinically relevant data, so we believed that fine-tuning the model on the development set could improve its performance on the test set. Because of time and hardware constraints, we randomly sampled 3000 instances from the 622,703 test set instances as unlabeled data for self-training. We used the same hyperparameters for fine-tuning the source-domain model on the development set and self-training the fine-tuned model on the randomly sampled test data. All the hyperparameters are shown in table 4 in appendix A.2. Our submission ranked 2<sup>nd</sup>. Table 2 shows that our system outperformed both baseline models provided by the organizers.

### 4.2 Time Expression Recognition

We did not fine-tune the source-domain model on the development set during the test phase. The development set is from the newswire domain, while the test set is from the food security domain. We thought that there might be a large difference between these two domains. Fine-tuning on a different domain may hurt the performance of the model on the test set. As with the code provided by the organizer, we used the sentencizer from Spacy (Honnibal et al., 2020) to split the input documents into sentences and used them as inputs to the model. All the hyperparameters are shown in table 5 in



#	Strategy	F	P	R
<i>Test Phase</i>				
1	SD (baseline)	0.794	0.849	0.746
2	SD + FT (baseline)	<b>0.804</b>	0.827	0.782
3	SD + AL (32*5) + DA (5) + Manual Annotations	0.795	0.783	0.807
<i>Post-Evaluation</i>				
4	SD + AL (32*5) + DA (5) + Manual Annotations (fixed seasonal(1y))	0.837	0.824	0.850
5	SD + AL (32*5) + DA (5) + Gold Annotations	0.955	0.945	0.965
6	SD + FT + AL (32*5) + DA (5) + Gold Annotations	<b>0.959</b>	0.949	0.969
7	SD + AL (32*5) + Gold Annotations	0.890	0.893	0.887
8	SD + AL (16*5) + DA + Gold Annotations	0.929	0.918	0.941
9	SD + AL (8*5) + DA + Gold Annotations	0.900	0.880	0.920
10	SD + AL (4*5) + DA + Gold Annotations	0.877	0.860	0.894
11	SD + AL (4*5) + Gold Annotations	0.851	0.846	0.855

Table 3: The performance of the time expression recognition systems during the test and post-evaluation phases. SD is the source-domain model. FT is fine-tuning on the development set. AL ( $k*i$ ) is active learning with  $k$  samples and  $i$  iterations. DA ( $n$ ) is data augmentation with  $n$  generated examples.

appendix A.2. Our submitted system ranked 6<sup>th</sup>. Table 3 shows that our submitted system’s performance (row 3) is no better than the best baseline model (row 2) provided by the organizers.

To investigate the reasons for the lower-than-expected test performance, we used the gold annotations in the test set for our post-evaluation runs (row 5-11 in table 3). Note that performance for these rows will be artificially inflated, since up to 160 of the 926 test sentences were included in the system’s training data. Nonetheless, we see that by using the gold annotations instead of our manual annotations (row 5 vs row 3 in table 3), the performance of our system improved by 0.160 F1 score. This seems to suggest that our system can improve its generalization ability if we can accurately label the target domain data.

We further analyze where the performance improvement comes from in section 5 and provide a detailed analysis of our annotation errors and give feedback on annotation guidelines in section 6.

## 5 Time Expressions Ablation Study

**Effect of Fine-Tuning on Dev Data** From the baseline models’ performances (row 1 vs row 2 in table 3), we can see that the test performance of the model fine-tuned on the development set is slightly better than the pure source-domain model (+.010 F1 score). To verify if this is also true for our active learning system, we add the fine-tuning strategy to

our system (row 6 in table 3) and run the system on the labeled portion of the test set. The results (row 5 vs row 6 in 3) indicate that fine-tuning on the additional domain continues to help a bit (+.004 F1 score) even when followed by active learning.

**Effect of Data Augmentation** We also investigate the contribution of our data augmentation strategy, removing it from our system and running on the labeled test set. The result shows that data augmentation brings a +.065 F1 score improvement to our system (row 7 vs row 5 in table 3). This indicates that data augmentation was a major source of performance improvements.

**Effect of Size of Annotation Data** In real-world use cases, we often want to keep the size of annotated data as small as possible, since annotation is time consuming and error-prone. To understand how our system performs with less manual annotated examples, we reduce the number of sentences to be annotated at each active learning iteration to 16, 8 and 4 resulting in rows 8, 9 and 10 in table 3. The results show that with only 20 correctly annotated sentences but incorporating data augmentation (row 10), our system outperform the best baseline model (row 2) by .073 F1. If we remove data augmentation from this model (row 11) its performance declines, but still outperforms the best baseline model by .047 F1.

## 6 Time Expressions Annotation Analysis

Though gold annotations led to large performance improvements, the annotation for this task is challenging for untrained people. Through reading the SCATE annotation guidelines and posing questions on the share task google group, our team annotated 160 sentences of which 48 sentences were in the labeled portion of the test set. We annotated 13008 tokens in total (including padding tokens) and our overall accuracy on the gold 48 sentences is 0.991 for all categories and 0.785 excluding the category O. We report detailed performance for each of the entity types in table 6 in appendix A.3.

We found several annotation patterns where our team consistently disagreed with the gold annotations. Our errors can be broadly attributed to two reasons: misinterpretation/underspecification of annotation guidelines, and ambiguity of the phrases.

**Errors from misinterpretations/underspecification of annotation schema** : We annotated the token *seasonal(ly)* (e.g., *seasonal progress*, *seasonal rainfall*) as Calendar-Interval instead of Season-Of-Year as we thought Season-Of-Year is applied to seasons that are explicitly specified (such as summer). We considered *seasonal* similar to *weekly*, both referring to an interval unspecified. However, Season-Of-Year could be applied to very broad categories such as dry seasons and rainfall seasons including seasons that are not specified. Also, *seasonal* unlike *weekly/monthly/yearly* only refers to one season of a year instead of every season of a year. Due to the ubiquity of this token in the dataset, this error affects our overall performance. Correcting the annotation of this particular token leads to +.042 F1 score improvement (row 4 vs row 3 in table 3). Another erroneous pattern is that we double-annotated the phrase such as *from ... to ...* and *between ... to ...*. Specifically, we annotated both adpositions instead of choosing the first adposition only. Finally, we also annotated more modifiers than the gold annotations. For instance, we annotated *marketing* in *marketing year* and *long* in *long dry Jiaal Season* as ‘Modifier’ instead of the category ‘O’. It turns out that the category of modifier in the gold annotation is a closed category, and only a specific set of tokens are considered modifiers.

**Errors from ambiguity within phrases** Some phrases allow multiple interpretations that lead to different ways of annotations. For instance, con-

fusion between ‘Period’ and ‘Calendar Interval’ occurred frequently (e.g., we annotated *weeks* in *recent weeks* as ‘Period’ rather than ‘Calendar-Interval’). Although “/” between seasons is commonly annotated as ‘I-Season-Of-Year’ in the gold annotations, we found different roles it might play in specific contexts. For example, if “/” is used between two terms that refer to the same season, then it should be annotated as ‘I-Season-Of-Year’; if it is used between two non-adjacent seasons, it should be annotated as ‘Union’; and if it is used between two adjacent seasons, then it could be annotated as ‘Between’ or ‘Union’. Thus, the correct annotation requires a surprising amount of external knowledge about Ethiopian season terms. In fact, there are still cases that remained uncertain: For example, *Xaran* refers to seasonal rains from April through September and *Xagaa* refers to the second dry season (July to September) and when the two tokens joined by “/” it is difficult to interpret the meaning of “/”. We also found the conjunction *and* causes ambiguity. For example, *and* in *rains in May and August* could be considered as an operator over months (i.e., rains in Union(May, August)) or an operator over rains (i.e., Union(rains in May, rains in August)). The former understanding requires annotating *and* whereas the latter does not despite the fact that the two interpretations are essentially semantically equivalent. Lastly, we also found the particle *the* is difficult to annotate. For instance, *the* in *the month* depending on the context may be annotated as *this* or *last* and sometimes the context may not be clear enough to tell the differences.

Our annotation analysis leads to several suggestions for the annotation schema and the documentation. Our errors in the first category indicate some potential helpful updates can be made such as including more examples in categories (e.g., ‘Season-Of-Year’), explicit documentation of whether the certain category is closed or open as well as the specific manner to deal with multi-word phrases or even circumpositions. The second category of errors, however, might involve the refinement of the annotation schema. For example, maybe ‘Between’ and ‘Union’ can be unified together, and ‘Period’ can be merged into ‘Calendar Interval’ or confined to an explicit set of circumstances.

## 7 Conclusion

Our overall rank (by F1 score) for negation detection task was 2<sup>nd</sup> and for time expression recogni-

tion was 6<sup>th</sup>.

Our results suggest that simple self-training can be used in sentence-level SFDA tasks to improve a trained model’s performance on a new domain. As for token-level tasks, our analysis shows that both active learning and data augmentation can bring significant performance improvements, but the premise is that the data in active learning can be correctly annotated. Our analysis and feedback could also be used to improve the SCATE annotation guidelines/schema in future work.

## Acknowledgments

Research reported in this publication was supported by the National Library of Medicine of the National Institutes of Health under Award Numbers R01LM012918 and R01LM010090. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

## References

- Steven Bethard and Jonathan Parker. 2016. [A semantically compositional annotation scheme for time normalization](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 3779–3786, Portorož, Slovenia. European Language Resources Association (ELRA).
- Minmin Chen, Zhixiang Xu, Kilian Q. Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on International Conference on Machine Learning, ICML’12*, page 1627–1634, Madison, WI, USA. Omnipress.
- David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1996. Active learning with statistical models. *J. Artif. Int. Res.*, 4(1):129–145.
- Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030.
- Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML’11*, page 513–520, Madison, WI, USA. Omnipress.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Egoitz Laparra, Steven Bethard, and Timothy A Miller. 2020. [Rethinking domain adaptation for machine learning over clinical language](#). *JAMIA Open*, 3(2):146–150.
- Egoitz Laparra, Xin Su, Yiyun Zhao, Ozlem Uzuner, Timothy Miller, and Steven Bethard. 2021. SemEval-2021 task 10: Source-free domain adaptation for semantic processing. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval)*.
- Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. [SemEval 2018 task 6: Parsing time normalizations](#). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 88–96, New Orleans, Louisiana. Association for Computational Linguistics.
- Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692.
- Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard S. Zemel. 2016. [The variational fair autoencoder](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. [Snippext: Semi-Supervised Opinion Mining with Augmented Data](#), page 617–628. Association for Computing Machinery, New York, NY, USA.
- Patrice Y. Simard, Dave Steinkraus, and John C. Platt. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2, IC-DAR ’03*, page 958, USA. IEEE Computer Society.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- David Yarowsky. 1995. [Unsupervised word sense disambiguation rivaling supervised methods](#). In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

## A Appendix

### A.1 Entity Candidates Pool

**Second-Of-Minute:** 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60.

**Day-Of-Month:** 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

**This:** today, these, this, the, now, current, These, This, The, Current.

**Minute-Of-Hour:** 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60.

**Year:** 1970, 1971, 1973, 1974, 1975, 1980, 1981, 1982, 1984, 1990, 1992, 1995, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2020.

**Month-Of-Year:** January, February, March, April, May, June, July, August, September, October, November, December, Jan, Feb, Mar, Apr, Aug, Sept, Sep, Oct, Nov, Dec, 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, may.

**Next:** later, future, following, next, coming, upcoming, Following.

**Hour-Of-Day:** 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 01, 02, 03, 04, 05, 06, 07, 08, 09.

**Time-Zone:** ART, CT, EGT, EST, MART, MMT, NT, TOT, WIT.

**Two-Digit-Year:** 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, tens, Tens.

**Calendar-Interval:** minute, minutes, day, days, daily, today, eve, week, weekly, weeks, months, month, monthly, quarter, quarterly, year, years, annual, annually, Daily, Eve, Month, Monthly, Year, Annual.

**Modifier:** more than, approx, less than, start, mid, middle, end, over, early, around, late, older, financial, fiscal, nearly, longer than, almost, at least, or so, about, beginning, More than, Approx, Less than, Mid, Middle, End, Over, Early, Around, Late, Nearly, At least, About, Beginning.

**Last:** before, last, latest, previously, recent, recently, the past, ever, previous, past, earlier, pre, Before, Last, Recent, Ever, Previous, Past, Earlier, Pre.

**Between:** from, since, until, between, From, Since, Until, Between.

**Day-Of-Week:** Tuesday, Wednesday, Mon, Tues, Tue, Wed, Sat, Sun.

**Period:** period, periods, week, months, minute, year, term, day, time, years, second, moment, minutes, long-term, decades, decade, short-term, month, weeks, days, Year, Term, Time, Second, Short-term, Month.

**Part-Of-Day:** night, Noon.

**Before:** previously, prior, before, ago, pre, by, earlier, next, Prior, Before, Ago, Pre, By, Earlier.

**NthFromStart:** second, first, fourth, third, seventh, Second, 3rd, 5th, 7th, 25th, 47th, 75th.

**After:** after, from, later, post, After, From, Post.

**Season-Of-Year:** winter, spring, summer, fall, season, autumn, Summer, Fall, Season.

**AMPM-Of-Day:** pm, am, PM, AM.

## A.2 Hyperparameters

Hyperparameter	Value
number of examples from the test set used for self-training	3000
maximum number of self-training iterations	30
actual number of self-training iterations	2
self-training threshold ( $\tau$ )	0.95
maximum sequence length	128
batch size	32
epochs	10
learning rate	5e-5
learning rate schedule type	linear
learning rate warm up steps	0
weight decay	0.0
maximum gradient norm	1.0

Table 4: Hyperparameters for negation detection system

Hyperparameter	Value
number of active learning iterations ( $i$ )	5
number of sentences to annotate at each active learning iteration ( $k$ )	32
number of new sequence to augment for each annotated time entity ( $n$ )	5
maximum sequence length	271
batch size	32
epochs	8
learning rate	3e-5
learning rate schedule type	linear
learning rate warm up steps	0
weight decay	0.0
maximum gradient norm	1.0

Table 5: Hyperparameters for time expression recognition system.

## A.3 Manual Annotation Performance

Type	F	P	R	# in gold annotations	# in our annotations
I-Calendar-Interval	0.000	0.000	0.000	0	5
I-Last	0.000	0.000	0.000	2	5
I-Between	0.000	0.000	0.000	1	0
I-Modifier	0.000	0.000	0.000	1	4
B-This	0.154	0.333	0.100	10	3
B-Before	0.222	0.250	0.200	5	4
B-Union	0.250	0.143	1.000	1	7
B-Period	0.400	0.308	0.571	7	13
B-After	0.500	1.000	0.333	9	3
I-Frequency	0.500	1.000	0.333	3	1
B-Modifier	0.519	0.389	0.778	9	18
I-Period	0.588	1.000	0.417	12	5
B-Last	0.615	0.800	0.500	8	5
B-Calendar-Interval	0.632	0.529	0.783	23	34
B-Intersection	0.667	1.000	0.500	2	1
B-Frequency	0.750	0.750	0.750	4	4
I-Number	0.769	1.000	0.625	8	5
B-Season-Of-Year	0.792	0.864	0.731	52	44
B-NthFromStart	0.800	0.667	1.000	2	3
B-Between	0.831	0.750	0.931	29	36
I-Season-Of-Year	0.896	0.972	0.831	83	71
B-Number	0.909	0.909	0.909	11	11
O	0.997	0.997	0.998	12627	12632
B-Year	1.000	1.000	1.000	36	36
B-Month-Of-Year	1.000	1.000	1.000	57	57
B-Part-Of-Day	1.000	1.000	1.000	1	1
B-Two-Digit-Year	1.000	1.000	1.000	4	4
B-Next	1.000	1.000	1.000	1	1

Table 6: Performance of each annotated entity types. Please note that when the number in gold annotation is 0, it means that we annotate this entity type, but it does not appear in the gold annotations (test labels).

# KnowGraph@IITK at SemEval-2021 Task 11: Building Knowledge Graph for NLP Research

Shashank Shailabh\* Sajal Chaurasia\* Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{shailabh, sajal}@iitk.ac.in

ashutoshm@cse.iitk.ac.in

## Abstract

Research in Natural Language Processing is making rapid advances, resulting in the publication of a large number of research papers. Finding relevant research papers and their contribution to the domain is a challenging problem. In this paper, we address this challenge via the SemEval 2021 Task 11: NLPContributionGraph, by developing a system for a research paper contributions-focused knowledge graph over Natural Language Processing literature. The task is divided into three sub-tasks: extracting contribution sentences that show important contributions in the research article, extracting phrases from the contribution sentences, and predicting the information units in the research article together with triplet formation from the phrases. The proposed system is agnostic to the subject domain and can be applied for building a knowledge graph for any area. We found that transformer-based language models can significantly improve existing techniques and utilized the SciBERT-based model. Our first sub-task uses Bidirectional LSTM (BiLSTM) stacked on top of SciBERT model layers, while the second sub-task uses Conditional Random Field (CRF) on top of SciBERT with BiLSTM. The third sub-task uses a combined SciBERT based neural approach with heuristics for information unit prediction and triplet formation from the phrases. Our system achieved F1 score of 0.38, 0.63 and 0.76 in end-to-end pipeline testing, phrase extraction testing and triplet extraction testing respectively.

## 1 Introduction

Given the advancements in Natural Language Processing (NLP), a large number of research papers are published every year. However, given the field's dynamic nature, keeping track of all the papers is a non-trivial task. This motivated the formulation

of an Open Research Knowledge Graph (Jaradeh et al., 2019), a knowledge graph of research contributions and the relation between them. Task 11 of SemEval 2021 (D'Souza et al., 2021) formalizes the building of a contributions-focused knowledge graph of NLP literature. The task is divided into three sub-tasks:

- **Sub-task A** Extracting sentences that posit contributions in a research paper.
- **Sub-task B** Extracting relevant phrases that include scientific terms and relational cues from the extracted sentences of the sub-task A.
- **Sub-task C** Triplet (subject phrase, predicate phrase, object phrase) formation from the extracted phrases of the sub-task B and classification of the triplet in one of the information units (IU). There are twelve information units (Research problem, Approach, Results, Model, Code, Dataset, Experimental setup, Hyperparameters, Baselines, Tasks, Experiments, and Ablation analysis), each focusing on different sections in a research paper. These information units can represent all the findings given in a research paper. Out of twelve, first three information units are present in each article.

Recently, transformer-based approaches have been popular for NLP applications (Liu and Lapata, 2019; Yao et al., 2019). For the sub-task A, we propose a sentence level classifier, leveraging SciBERT (Beltagy et al., 2019) and Bidirectional Long Short-Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997). SciBERT model has been trained on 1.14M scientific papers from Semantic Scholar corpus, which has 18% papers from the computer science domain. Our system for sub-task B also uses SciBERT based model using CRF

\* Authors equally contributed to this work.

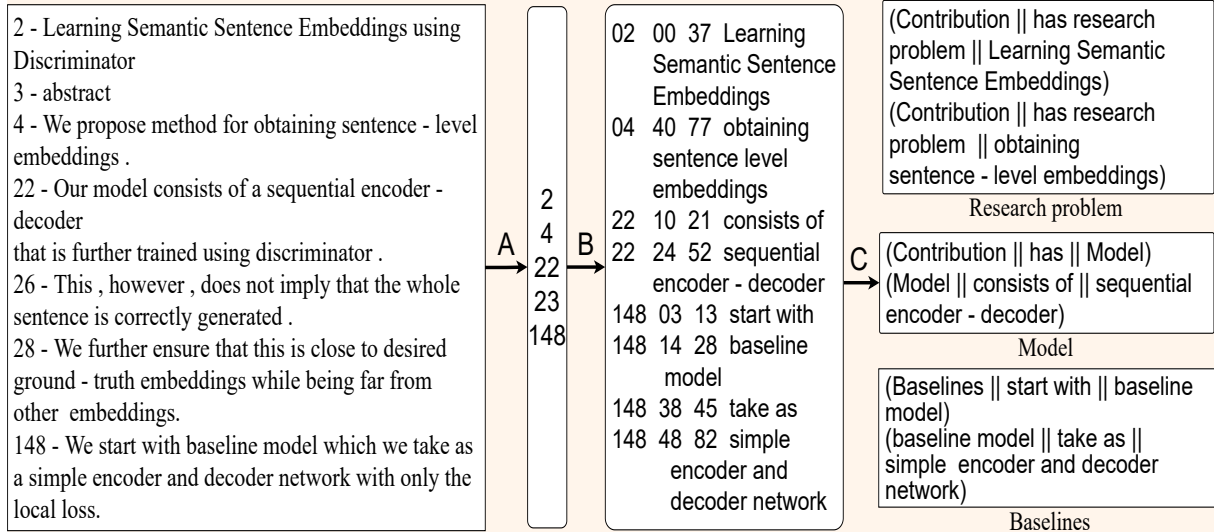


Figure 1: Dataset Example

(Lafferty et al., 2001) on top of BiLSTM layers using BILUO (B=start token of phrase, I=interior tokens of phrase, L=last token of phrase, U=single token phrase and O=Non-phrase tokens) labelling scheme for tokens. For sub-task C, we build a combination of neural and heuristic-based approach. The IU prediction for sub-task C is at document level where two information units use heuristic approach while others use a multi-label classifier based on BiLSTM stacked on top of SciBERT. For triplet formation in sub-task C, we use a separate SciBERT+BiLSTM classifier along with heuristics. These triplets are classified into one of the already predicted information units, i.e., the output unit having the maximum score among the predicted IU. However, some of the IU, such as Baselines, Ablation analysis, Code and Research problem triplets, perform well using heuristic, so we use heuristic instead of a neural approach for these IU. Our proposed system was ranked third in the overall end-to-end pipeline (A+B+C) testing and achieved F1 score of 0.38. Our proposed model was ranked fourth in phrase extraction ( $A_{GT}+B+C$ ) and triplet extraction testing ( $A_{GT}+B_{GT}+C$ ) with a F1 score of 0.63 and 0.76 respectively where  $A_{GT}$  and  $B_{GT}$  represent ground-truth for sub-task A and B respectively. Phrase extraction testing uses ground-truth labels for sub-task A, while triplet extraction testing uses ground-truth for both sub-task A and B. We found that the heuristic-based model for two IU (Research problem and Code) in sub-task C can achieve high performance and achieved F1 score of 0.98 and 1.00, respectively. Our system imple-

mentation code is made available via GitHub<sup>1</sup>.

## 2 Background

### 2.1 Problem Definition

Consider a document  $D = \{s_1, s_2, \dots, s_i, \dots, s_N\}$  having  $N$  sentences  $s_i$ . Sub-task A finds  $M$  contribution sentences denoted by  $S = \{s_1, s_2, \dots, s_M\}$  from  $D$ . Sub-task B selects phrases  $P = \{p_1, p_2, \dots, p_i, \dots, p_L\}$  where  $p_i$  is a phrase selected from a sentence  $s \in S$  and  $L$  is total number of phrases in  $D$ . Sub-task C is forming triplets of extracted phrases for IU denoted by  $U = \{u_1, \dots, u_i, \dots, u_X\}$  where  $u_i$  is one of the twelve IU and  $X$  is number of IU in document  $D$  ranging between three to twelve. For each  $u_i \in U$ , there is a triplet set called  $T^i = \{(su_1^i, pr_1^i, ob_1^i), (su_2^i, pr_2^i, ob_2^i), \dots, (su_j^i, pr_j^i, ob_j^i), \dots, (su_O^i, pr_O^i, ob_O^i)\}$  where  $(su_j^i, pr_j^i, ob_j^i)$  is a triplet representing subject, predicate (relation) and object respectively and  $O$  is total number of triplets in  $u_i$  IU in document  $D$ . An example dataset is given in Figure 1 for reference. Here, on moving from left to right is research paper, contribution sentences, phrases and IU along with triplets given in a research paper respectively.

### 2.2 Related Work

Knowledge graphs (Rebele et al., 2016; Hertling and Paulheim, 2018; Lehmann et al., 2015; Carlson et al., 2010) have shown to be helpful in several areas such as search, knowledge extraction, inter

<sup>1</sup><https://github.com/sshailabh/SemEval-2021-Task-11>



alia. However, only a handful are based on research articles (Xu et al., 2020). Typically, most knowledge graphs are created with rule-based approaches, hence, limiting their performance and generalization. However, some recent approach such as Sang et al. (2018); Wang et al. (2020b) uses neural approach in biomedical literature. To the best of our knowledge, there is no available contributions-focused knowledge graph over NLP literature using the neural approach.

**Sub-task A:** The sub-task of extracting contribution sentences can also be posed as an extractive summarization problem (Nallapati et al., 2016; Narayan et al., 2018; Liu, 2019; Cheng and Lapata, 2016; Zhou et al., 2018; Dong et al., 2018; Wang et al., 2020a). BERTSUM (Liu and Lapata, 2019) and MATCHSUM (Zhong et al., 2020) are the recent methods leveraging language models and uses ROUGE-1, ROUGE-2 and ROUGE-L scores (Lin, 2004) on DailyMail data-set (Hermann et al., 2015). However, this extractive summarization technique may not be applicable in our case due to a number of reasons. Firstly, extractive summarization alone will not give all the contribution sentences because some sentences may not be relevant to the summarization task. Secondly, extractive summarization models are not tested on large documents such as research articles due to the limitation of the input token length for transformer-based language models. Some long document transformer-based methods are proposed (e.g., Beltagy et al., 2020), and can consider documents up to a length of 4096 tokens, however, in our case, documents have on an average  $\sim 10,000$  tokens. Some of the extractive summarization methods (Liu and Lapata, 2019; Miller, 2019) take the number of contribution sentences as a hyper-parameter, but in our case, this is a trainable parameter in our model.

**Sub-task B:** Sub-task B closely resembles the phrase extraction problem and several neural methods (Zhu et al. (2020); Wang et al. (2016); Zhang et al. (2016), inter alia) and non-neural based methods (using n-grams and noun-phrases with certain Part-of-speech (POS) patterns (Hulth, 2003)) have been proposed. Gollapalli et al. (2017) have shown that CRF has the potential to improve the existing phrase extraction model. Alzaidy et al. (2019) jointly leverages CRF and BiLSTM to capture hidden semantics for phrase extraction. Zhu et al. (2020) extended the work of Alzaidy et al. (2019) with the idea of self-training and used word em-

beddings, POS embeddings, and dependency embeddings with a BILUO labelling scheme in the output. Our proposed model took inspiration from Zhu et al. (2020) and propose a SciBERT based model using CRF on top of BiLSTM layers using BILUO labelling scheme on tokens. Our model captures better semantics than the word embeddings based approach in Zhu et al. (2020) because of SciBERT, which is trained on the scientific corpus. Moreover, our model uses the WordPiece tokenizer and hence, robust to Out-of-Vocabulary (OOV) tokens. Sahrawat et al. (2020) used contextual embeddings to the BiLSTM and CRF model using BIO (B=start token of phrase, I=continuation tokens of phrase and O=Non-phrase tokens) labelling scheme. Ratnov and Roth (2009) discussed that the BILUO scheme is superior to the BIO scheme; hence we adopt the BILUO scheme for sub-task B. Recently, Lai et al. (2020) combined sequence labelling with joint learning inspired from self-distillation to boost model performance on unsupervised datasets. However, their model used BIO labelling scheme and gave a comparable or marginal improvement in a supervised setting.

**Sub-task C:** The sub-task C can be divided into two parts - information units (IU) prediction and triplet formation. The information unit prediction and triplet formation have been approached in the literature mainly using rule-based methods. Rusu et al. (2007) suggests using syntactic parsers for generating parse trees, followed by triplet extraction using parser dependent techniques. Jivani et al. (2011) proposed an algorithm that exhibits the relationship between subject and object in a sentence using Stanford parser. This rule-based algorithm can form multiple triplets from a sentence as compared to Rusu et al. (2007). Stanford OpenIE Relation triplet formation (Angeli et al., 2015) uses a classifier, which learns to extract self-contained clauses from longer sentences to form the final triplets using heuristics. Hamoudi (2016) and Jaiswal and George (2015) are also rule-based methods for triplet formation using Stanford dependency parser and constituency parser, respectively. KG-Bert (Yao et al., 2019) uses the BERT language model and utilize entity and relation descriptions of a triplet to compute its scoring function.

### 3 System Overview

The proposed system is shown in Figure 2 depicting the entire pipeline and its respective model.

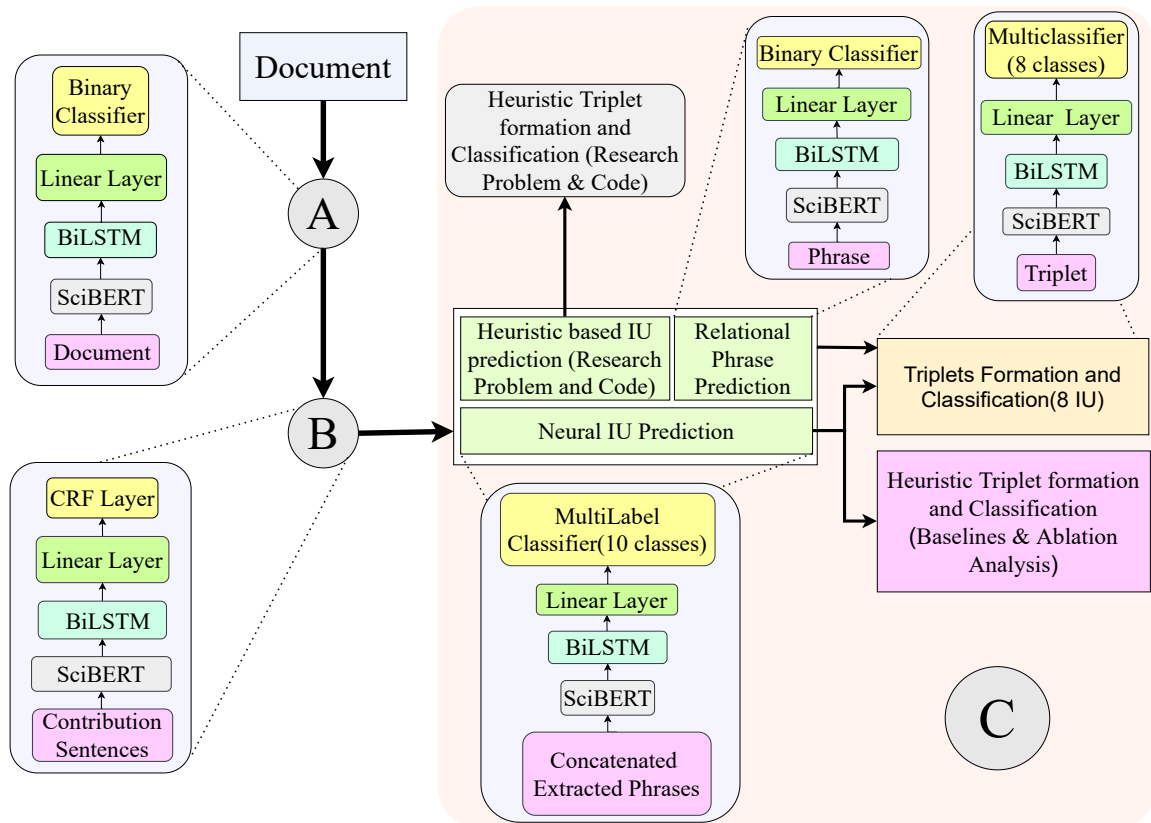


Figure 2: Overall system architecture for end-to-end pipeline showing all the sub-tasks and their respective model.

### 3.1 Sub-Task A

**Initial Experimentation:** We experimented with SciBERT, a language model trained on research papers to build a binary classifier. The classifier encodes the sentence using feature representation corresponding to [‘CLS’] token from the last layer of the pre-trained SciBERT model to predict a binary label. Since the fine-tuning dataset is small, linear layers were not able to capture contextual information well. We experimented by adding a Convolution Neural Network (Zhang and Wallace, 2015) layer on the top of SciBERT. The CNN architecture use three kernels of size two, three and four. The model boosted the performance; however, the model cannot classify long sentences due to their ambiguous nature and lack of CNN’s capacity to capture long semantic dependency among tokens.

**Proposed Approach:** We propose the SciBERT+BiLSTM model (a sentence-level binary classifier where BiLSTM is stacked on top of the SciBERT model). This helps to encode hidden semantics and long-distance dependency. Consider the training dataset as  $Tr = \{D_1, D_2, \dots, D_i, \dots, D_Z\}$  comprising of  $Z$  documents. Each  $D_i$  can be represented as  $D_i = \{s_{i1}, s_{i2}, \dots, s_{ij}, \dots, s_{iN}\}$  where  $N$

is the number of sentences in the document and  $s_{ij}$  is the  $j^{th}$  sentence of document  $D_i$ . Each sentence is assigned a ground-truth label where label “1” represents a contribution sentence and label “0” a non-contribution sentence. The sentences are processed using a SciBERT model followed by stacked BiLSTM layers whose output is further processed through linear layers with ReLU (Nair and Hinton, 2010) non-linearity. We add dropout layers to avoid overfitting. The last linear layer consists of two units corresponding to label “0” and label “1”. The final output label is the label whose corresponding unit has a higher score in the last linear layer. Our loss function is weighted binary cross-entropy loss, where weights are in according to the number of samples in each class.

### 3.2 Sub-Task B

**Initial Experimentation:** We built our initial method, the BiLSTM+CRF model, on the lines of Zhu et al. (2020). The model uses a BiLSTM layer along with CRF for sequence to sequence (BILUO) labelling in order to mark the phrases in a sentence. We introduced SciBERT (by replacing BiLSTM layers) to fine-tune and better generalise with an increase in performance. We tested the significance

of CRF by replacing it with a softmax layer, which gave poor performance since it is unable to learn the constraints in the BILUO scheme.

**Proposed Approach:** We further improved the SciBERT+CRF model to improve semantic information. Our proposed model stack BiLSTM layers on top of SciBERT, followed by CRF (see Figure 2). The word-level representation  $\{x_1, x_2, \dots, x_N\}$  from the input sentence passes through the SciBERT tokenizer. We used the representation of the first sub-token for every word as the input to the SciBERT. The tokenized input is passed into the SciBERT layer, followed by the BiLSTM layer. The final feature output is mapped to a hidden linear layer to get the score matrix  $Z$ , which is passed into the CRF layer for label prediction  $y$ . The CRF layer is the same as described in Lample et al. (2016).

The output produced by the SciBERT + BiLSTM + Hidden Layer corresponds to a scoring matrix  $Z^{(n \times l)}$  where  $n$  denotes the number of words in the input sentence, and  $l$  is the number of labels ( $l=5$ ). The score of an output sequence  $y$  using CRF is given by:

$$Scr(s, y) = \sum_{i=0}^n (Z_{i, y_i} + T_{y_{i-1}, y_i}) \quad (1)$$

where  $Z_{i,j}$  denotes the score of word  $w_i$  with the  $j$ th label,  $T_{y_{i-1}, y_i}$  is the transition score from the label  $y_{i-1}$  to  $y_i$ ,  $y = \{y_1, y_2, \dots, y_n\}$  is the sequence of true labels and  $Scr(s, y)$  corresponds to output score for sentence  $s$  and true labels  $y$ . A softmax over all possible label sequences yields a probability for true labels  $y$ :

$$P(y|s) = \frac{\exp(Scr(s, y))}{\sum_{y' \in Y(s)} \exp(Scr(s, y'))} \quad (2)$$

where  $Y(s)$  corresponds to all the possible label sequences for sentence  $s$ . Now during training, our task is to maximize the log-probability of the correct label sequence  $y$ . Model loss is defined as follows:

$$L(\Theta) = -(1/M) \sum_{i=1}^M \log(P(y_i|s_i)) + \frac{\lambda}{2} \|\Theta\|^2 \quad (3)$$

where  $s_i$  is the input sentence,  $y_i$  is the corresponding true label sequence,  $\Theta$  denotes the model parameters,  $\lambda$  is the regularization hyperparameter and  $M$  is the train set size. Output label prediction is made by:

$$y^* = \operatorname{argmax}_{y' \in Y(s)} P(y'|s) \quad (4)$$

Here  $y^*$  represents the final output label sequence,  $s$  is the input sentence,  $Y(s)$  is the set of possible label sequences and  $P(y'|s)$  denotes the probability of getting  $y'$  label sequence from sentence  $s$ . The phrases are extracted using BILUO scheme based on the prediction outputs.

### 3.3 Sub-Task C

**Initial Experimentation:** We used a combination of neural and rule-based approach for sub-task C. An IU triplet has three phrases - subject, predicate and object. Research problem and Code IU triplets have fixed subject and predicate in triplets. We employed a heuristic to scan the phrases of the first thirty lines of each document and select only those sentence's phrases that have only a single phrase extracted out. These phrases form object in Research problem IU triplets. A regex expression is used to extract all the sentences in the article that contain any URLs for Code IU triplet's object. However, only those URL sentences are selected which have token such as "our" or "code" or "our code".

Our initial approach was to form the triplets for all other IU and classify them into one of the ten remaining information units using SciBERT + BiLSTM multi-class classifier (BiLSTM layers stacked on top of SciBERT). The heuristic for triplet formation is based on orthographic visualization of the document - firstly, the phrases are arranged in the exact order as they appear in the original sentence. Then, every three consecutive pair of phrases present within the same sentence are considered as one triple. This approach gave us decent results since most of the research paper is written in the active voice; hence the subject phrase should occur first, then its corresponding predicate and last should be the object phrase. The SciBERT + BiLSTM multi-class classifier takes concatenated triplets as the input and their corresponding information unit as the ground truth. The loss is a cross-entropy loss, with the final softmax layer having ten classes corresponding to 10 information units. One of the drawbacks of the model is that the IU prediction depends on triplet formation's correctness. Further, a single triplet does not have enough context to be correctly classified into the correct information unit. We visualized triplet formation for feature extraction and found that some IU triplet's, such as Baselines and Ablation analysis, can be better formed using heuristics. Our proposed approach

is a better triplet formation model and eliminates these limitations.

**Proposed Approach:** Our proposed approach has some information unit such as Research problem and Code, whose model for prediction and triplet formation is entirely heuristic-based and the same as initial experimentation. Our proposed method for the rest of ten IU is divided into two parts -

**IU Prediction** - We propose a SciBERT+BiLSTM multi-label classifier (BiLSTM layers stacked on top of SciBERT) (refer to Figure 2) whose input is the concatenated phrases and predicts the IU of the document. The concatenation is in the order of the occurrence of phrases in the document. Moreover, these concatenated extracted phrases represent the whole document since it includes all relevant keywords of the research article necessary for information unit prediction. Hence, our proposed model encodes information at the document level, which makes it superior to the initial experimentation method.

**Triplet Formation and Classification** - In general, a phrase is either a relational phrase or a scientific term (Figure 1). A specific trend observed in ground truth triplets is that a triplet’s predicate is unique for a triplet. This contrasts with the subject and object phrase, which can be used multiple times in other triplets. Hence, a one-to-one relation between predicate phrase and triplet exists. We identify all predicate phrases from the extracted phrases of sub-task B; then, a corresponding triplet will be formed for each predicate phrase. We train a SciBERT+BiLSTM based binary classifier (BiLSTM layers stacked on top of SciBERT model) to identify the phrases which act as predicates (relational phrases). The model is fine-tuned on our dataset and uses the weighted binary cross-entropy loss. In labelling, “1” denote as predicate while “0” denote as non-predicate. To form triplets, we use a simple heuristic to arrange the phrases in the exact order as they appear in the original sentence. For every phrase predicted as a predicate, we take its previous phrase as the subject and its next phrase as the triplet object.

Now, a multi-class classifier for triplet classification for 8 IU (corresponding to all IU except Research problem, Code, Baselines and Ablation analysis) (refer to Figure 2) is built, which is similar to the one described in the initial experimentation. Since we have already predicted the information units, during inference, the triplet can be assigned

only to one of the already predicted information units, i.e., the output unit having the maximum score among the predicted IU.

We used rule-based heuristics for triplet formation of Baselines and Ablation analysis IU. The target sentences, whose phrases belong to baselines IU, are identified by selecting all the headings (i.e. lines having no punctuation) with words such as “baseline”, “comp” using a regex expression. Then, we took all the sentences between selected headings and their consecutive headings as the target sentences. The phrases associated with extracted sentences are used for triplet formation via the rule of three consecutive phrases present within the sentence. The same method is followed for Ablation analysis IU triplets with only change that the relevant headings are found using the regex expression that identifies if that heading contains the word such as “ablation”, “analysis”.

## 4 Experimental Setup

### 4.1 Data

The dataset annotation scheme is as per [D’Souza and Auer \(2020\)](#). The pre-processed dataset con-

Token Length	% of sentences less than token length
50	94.57
100	99.74
150	99.93
200	99.96

Table 1: Token length statistics on train set

sists of 287 annotated NLP research documents in the English language with ground truth for each sub-task. Train, dev and test set have 237, 50 and 155 documents, respectively. We have chosen 100 as the maximum token length in a sentence with WordPiece tokenizer since 99.7% sentences in the train set have less than or equal to 100 tokens. The Table 1 shows token length and percentage(%) of sentences less than that length. In the sub-task C, if there is no suitable predicate available in the extracted phrases, then the triplet’s predicate is chosen from the predefined set of predicates, i.e. “has”, “on”, “by”, “for”, “has value”, “has description”, “based on”, “called”. Table 2 shows the dataset statistics related to sub-task A and B. The dataset statistics for sub-task C is given in Table 3. Tasks information unit has no triplets in train set while

Statistics	Train	Dev
# Documents	237	50
# Contribution sentences	5096	1032
# Non-contribution sentences	50105	10451
# Avg. Sentences in doc.	232.915	229.7
# Avg. Tokens in sentence	20.622	21.06
# Avg. Contribution Sentences in doc.	21.38	20.24
# Avg. Phrases in doc.	128.53	92.52
# Avg. IU in doc.	4.43	4.46
# Max Tokens in sentence	396	193

Table 2: Dataset Statistics

# Information Unit Triplets	Train	Dev
Research problem	635	164
Approach	529	233
Model	3548	570
Code	40	9
Dataset	240	8
Experimental setup	1928	302
Hyperparameters	2267	254
Baselines	1625	146
Results	4989	657
Tasks	0	277
Experiments	1472	149
Ablation analysis	1407	155

Table 3: Number of triplets in each Information Unit on train and dev set.

Code and Dataset information units have very few triplets in dev set.

## 4.2 Hyperparameters

We have used the dev set to tune our hyperparameters. In every neural model, we are fine-tuning SciBERT.<sup>2</sup> We tried different batch sizes and learning rates for fine-tuning (Dodge et al., 2020). We found the best results using the AdamW optimizer in the neural models.

**Sub-task A** : We used batch size = 32, learning rate = 1e-05, epoch = 2, two layers of BiLSTM with hidden dimension of 400 and three linear layers (size = 800, 400, 100) with dropout = 0.1. Over-sampling of minority class (contribution sentences) counters the skewness in data.

**Sub-task B** : We used batch size = 1, learning rate = 2e-05, epoch = 4, single layer of BiLSTM with

<sup>2</sup><https://github.com/huggingface/transformers>

hidden dimension = 200 and linear layer with CRF. **Sub-task C** : We used batch size = 4, max-tokens = 512, learning rate = 2e-05, epoch = 16, threshold on sigmoid output = 0.5, two layers of BiLSTM with hidden dimension of 400 and three linear layers (size = 800,400,100) with dropout = 0.2 for multi-label classification of information units (ten out of twelve). We have used batch size = 32, max tokens = 25, learning rate = 2e-05, epoch = 4, two BiLSTM layers with hidden dimension = 400 and three linear layers (size = 800, 400, 100) with dropout = 0.1 for relational phrase prediction model. We have used batch size = 16, max tokens = 50, learning rate = 2e-05, epoch = 2, two BiLSTM layers with hidden dimension = 400 and three linear layers (size = 800, 400, 100) with dropout = 0.2 for triplet classification (eight out of twelve info-units).

## 4.3 Evaluation Metrics

In this task, organizers used Precision, Recall and F1 score metrics. In sub-task A, the predicted and ground-truth contribution sentences of the document calculate the metrics score. The sub-task B output has predicted phrase, contribution sentence number, starting and ending character number of the predicted phrase (Figure 1). These outputs are the basis for sub-task B metric calculations. The sub-task C has two groups of metrics. First is the Information Units prediction of a document (regardless of triplets). The second group calculates using both Information Units and triplets (the predicted triplet is correct only if it exactly matches the ground truth triplet and the ground-truth IU; otherwise, it is incorrect). The final score is the average of all the four F1 scores on the dataset. The participating team rankings are according to this score.

## 5 Results

Table 4 shows all the participating teams average F1 score. In end-to-end pipeline testing, the input is the documents. In phrase extraction testing ( $A_{GT}+B+C$ ), the input is the document and ground-truth for sub-task A. Here,  $A_{GT}$  represents the true label of sub-task A, and the F1 score for sub-task A is 1.00. In the triplet extraction phase ( $A_{GT}+B_{GT}+C$ ), the input is the document and ground truth labels (F1 score = 1.00) for both sub-tasks A and B to the system. Our team “Know-Graph@IITK” achieved an F1 score of 0.3783 and ranked third in end-to-end pipeline testing.

Team Name	A+B+C	A <sub>GT</sub> +B+C	A <sub>GT</sub> +B <sub>GT</sub> +C
BioNLP@UIUC	0.3828	<b>0.7612</b>	<b>0.8594</b>
ecnuica	0.3335	0.7113	0.8145
ITNLP	<b>0.4703</b>	0.6863	0.7931
<b>KnowGraph@IITK</b>	<b>0.3783</b>	<b>0.6318</b>	<b>0.7600</b>
INNOVATORS	0.3205	0.5252	0.5971
DuluthGrad	0.2838	0.4921	0.7579
YNU-HPCC	-	0.4562	0.6541
DFKI-SLT	0.2651	-	0.7137
NLP_IITGN	-	0.3522	-

Table 4: Average F1 score on test set of participating teams in end-to-end pipeline testing(A+B+C), phrase extraction testing(A<sub>GT</sub>+B+C) and triplet extraction testing(A<sub>GT</sub>+B<sub>GT</sub>+C).

In phrase extraction and triplet extraction testing phases, our system ranked fourth with an F1 score of 0.6318 and 0.76, respectively. Our heuristic-based triplet extraction for Code and Research problem information unit achieved excellent performance with an F1 score of 1.00 and 0.9756, respectively, on the test set.

### 5.1 Ablation and Error Analysis

Methods	Dev F1 score
Sub-task A	
SciBERT + CNN	0.440
SciBERT + BiLSTM	<b>0.451</b>
Sub-Task B	
BiLSTM + CRF	0.361
SciBERT + CRF	0.444
SciBERT + BiLSTM + CRF	<b>0.480</b>

Table 5: F1 score of several methods of sub-task A and sub-task B on development set.

We built several models using SciBERT, LSTM and CNN for each sub-task to understand each method’s significance (Table 5). We found that language models are knowledge-rich and boost the existing models. On top of language models, BiLSTM based model performs better than the CNN-based model due to long semantic dependency in sequential models. In sub-task B, BiLSTM+CRF based model performed inferior to the same model built on top of SciBERT. In triplet formation in sub-task C, our rule-based approach of Research problem and Code information unit yield excellent results (highest on the leaderboard). A significant improvement in the F1 score for Baseline and Ablation Analysis IU suggests that the rule-based approach can boost neural models since specific

patterns are present for these information units’ triplets. In sub-task A, the dataset is highly skewed between minority and majority classes (1:10), making the training of a neural model difficult. On visualization of sub-task B outputs, we found some ambiguous phrases that our model fails to predict correctly. Extracting both scientific and relation cue phrases with high precision and recall in a single model is difficult. Sometimes, our model predicts scientific phrases correctly but fails to predict relation cue phrase in the same sentence.

In sub-task C, some IU triplet’s such as Model, Hyperparameters, Results were present in large number comparatively and while Tasks and Dataset IU triplets are scarce in number. This skewness resulted in biasing of our multi-label classification model for IU prediction. In triplet formation, our predicate approach fails when the predicate is not present in the sentence and selected from the organizer’s closed set of predicates. The proposed system also fails in the case of branching between the triplets, i.e. multiple triplets share the same subject phrase. Our triplet formation is unable to predict Approach, Dataset and Tasks triple. On visualizing, we found that Model triplets are predicted instead of Approach triplets. Further, the triplets lack enough features to be classified into an information unit using a neural-based method. Our system performance on sub-task C is poor in the end-to-end pipeline and phrase extraction testing because we submitted our initial experimentation model.

## 6 Conclusion

In this paper, we have presented our system for NLPContributionGraph task of SemEval 2021. We found that neural models combined with heuristics

can build a knowledge graph by dividing it into small tasks. The heuristic-based model can outperform the neural approach in the triplet formation of some Information Units. In future work, neural models can use sparse transformers to encode long documents without increasing much memory. The pre-defined predicate incorporation could also be a future direction of work for our system.

## Acknowledgments

We thank Shubham Kumar Nigam for his guidance and SemEval-2021 organizers for running the competition. We are also very grateful to the Pytorch developers.

## References

- Rabah Alzaidy, Cornelia Caragea, and C Lee Giles. 2019. Bi-lstm-crf sequence labeling for keyphrase extraction from scholarly documents. In *The world wide web conference*, pages 2551–2557.
- Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. [Leveraging linguistic structure for open domain information extraction](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Andrew Carlson, Justin Betteridge, Richard C Wang, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 101–110.
- Jianpeng Cheng and Mirella Lapata. 2016. [Neural summarization by extracting sentences and words](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.
- Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.
- Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. 2018. [BanditSum: Extractive summarization as a contextual bandit](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium. Association for Computational Linguistics.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: Nlpcontributiongraph - structuring scholarly nlp contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Jennifer D’Souza and S. Auer. 2020. Nlpcontributions: An annotation scheme for machine reading of scholarly contributions in natural language processing literature. *ArXiv*, abs/2006.12870.
- Sujatha Das Gollapalli, Xiao-Li Li, and Peng Yang. 2017. Incorporating expert knowledge into keyphrase extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.
- Yassine Hamoudi. 2016. Extracting rdf triples using the stanford parser.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching machines to read and comprehend](#). In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems* 28, pages 1693–1701. Curran Associates, Inc.
- Sven Hertling and Heiko Paulheim. 2018. Dbkwik: A consolidated knowledge graph from thousands of wikis. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 17–24. IEEE.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Anette Hulth. 2003. [Improved automatic keyword extraction given more linguistic knowledge](#). In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 216–223.
- A. Jaiswal and V. George. 2015. A modified approach for extraction and association of triplets. *International Conference on Computing, Communication and Automation*.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Ed-dine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture*, pages 243–246.

- Ms Anjali Ganesh Jivani, Ms Amisha Hetal Shingala, and Paresh V Virparia. 2011. The multi-liaison algorithm. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 2(5).
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- Tuan Lai, Trung Bui, Doo Soon Kim, and Quan Hung Tran. 2020. [A joint learning approach based on self-distillation for keyphrase extraction from scientific documents](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 649–656, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#).
- Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. Dbpedia—a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195.
- Chin-Yew Lin. 2004. [ROUGE: A package for automatic evaluation of summaries](#). In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.
- Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.
- Yang Liu and Mirella Lapata. 2019. [Text summarization with pretrained encoders](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.
- Derek Miller. 2019. Leveraging bert for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.
- Vinod Nair and Geoffrey E. Hinton. 2010. [Rectified linear units improve restricted Boltzmann machines](#). In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, Haifa, Israel. Omnipress.
- Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *arXiv preprint arXiv:1611.04230*.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. [Ranking sentences for extractive summarization with reinforcement learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana. Association for Computational Linguistics.
- Lev Ratinov and Dan Roth. 2009. [Design challenges and misconceptions in named entity recognition](#). In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.
- Thomas Rebele, Fabian Suchanek, Johannes Hoffart, Joanna Biega, Erdal Kuzey, and Gerhard Weikum. 2016. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *International semantic web conference*, pages 177–185. Springer.
- Delia Rusu, Lorand Dali, Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. 2007. Triplet extraction from sentences. In *Proceedings of the 10th International Multiconference “Information Society-IS*, pages 8–12.
- Dhruva Sahrawat, Debanjan Mahata, Haimin Zhang, Mayank Kulkarni, Agniv Sharma, Rakesh Gosangi, Amanda Stent, Yaman Kumar, Rajiv Ratn Shah, and Roger Zimmermann. 2020. Keyphrase extraction as sequence labeling using contextualized embeddings. In *European Conference on Information Retrieval*, pages 328–335. Springer.
- Shengtian Sang, Zhihao Yang, Lei Wang, Xiaoxia Liu, Hongfei Lin, and Jian Wang. 2018. Sematyp: a knowledge graph based literature mining method for drug discovery. *BMC bioinformatics*, 19(1):1–11.
- Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuanjing Huang. 2020a. [Heterogeneous graph neural networks for extractive document summarization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219, Online. Association for Computational Linguistics.
- Minmei Wang, Bo Zhao, and Yihua Huang. 2016. Ptr: phrase-based topical ranking for automatic keyphrase extraction in scientific publications. In *International Conference on Neural Information Processing*, pages 120–128. Springer.
- Qingyun Wang, Manling Li, Xuan Wang, Nikolaus Parulian, Guangxing Han, Jiawei Ma, Jingxuan Tu, Ying Lin, Haoran Zhang, Weili Liu, et al. 2020b. Covid-19 literature knowledge graph construction and drug repurposing report generation. *arXiv preprint arXiv:2007.00576*.
- J. Xu, S. Kim, M. Song, M. Jeong, D. Kim, J. Kang, J. F. Rousseau, X. Li, W. Xu, V. I. Torvik, Y. Bu, C. Chen, I. A. Ebeid, D. Li, and Y. Ding. 2020. Building a PubMed knowledge graph. *Sci Data*, 7(1):205.
- Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. [KG-BERT: BERT for Knowledge Graph Completion](#). *arXiv e-prints*, page arXiv:1909.03193.



- Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. [Keyphrase extraction using deep recurrent neural networks on Twitter](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 836–845, Austin, Texas. Association for Computational Linguistics.
- Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners’ guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.
- Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. [Extractive summarization as text matching](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.
- Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. 2018. [Neural document summarization by jointly learning to score and select sentences](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne, Australia. Association for Computational Linguistics.
- X. Zhu, C. Lyu, D. Ji, H. Liao, and F. Li. 2020. Deep neural model with self-training for scientific keyphrase extraction. *PLoS One*, 15(5):e0232547.

# YNU-HPCC at SemEval-2021 Task 11: Using a BERT Model to Extract Contributions from NLP Scholarly Articles

Xinge Ma, Jin Wang and Xuejie Zhang  
School of Information Science and Engineering  
Yunnan University  
Kunming, China

Contact: maxinge@mail.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

## Abstract

This paper describes the system we built as the YNU-HPCC team in the SemEval-2021 Task 11: *NLPContributionGraph*. This task involves first identifying sentences in the given natural language processing (NLP) scholarly articles that reflect research contributions through binary classification; then identifying the core scientific terms and their relation phrases from these contribution sentences by sequence labeling; and finally, these scientific terms and relation phrases are categorized, identified, and organized into subject-predicate-object triples to form a knowledge graph with the help of multiclass classification and multi-label classification. We developed a system for this task using a pre-trained language representation model called BERT that stands for Bidirectional Encoder Representations from Transformers, and achieved good results. The average  $F_1$ -score for Evaluation Phase 2, Part 1 was 0.4562 and ranked 7th, and the average  $F_1$ -score for Evaluation Phase 2, Part 2 was 0.6541, and also ranked 7th.

## 1 Introduction

As the number of research publications increases, there is a growing need for digital libraries to equip researchers with alternative knowledge representations. In addition, because scientific literature is growing at a rapid rate and researchers today are faced with a publication deluge, it is difficult to keep up with the research progress even within ones own narrow discipline. The open research knowledge graph (ORKG) (Jaradeh et al., 2019) is posited as a solution to the problem of keeping track of research progress without the cognitive overload imposed by reading dozens of full papers. To this end, the aim of this task is to build a comprehensive knowledge graph that represents the research contributions of scholarly publications

per paper and also shows where the contributions are interconnected across papers (D’Souza and Auer, 2020).

The task was defined on a dataset containing natural language processing (NLP) scholarly articles with their contributions structured to be integrable within a knowledge graph infrastructure, such as the ORKG. The structured contribution annotations were provided as follows: (1) contribution sentences: a set of sentences about the contribution in the article; (2) scientific terms and relations: a set of scientific terms and relational cue phrases extracted from the contribution sentences; and (3) triples: semantic statements that pair scientific terms with a relation, modeled toward the subject-predicate-object statements for building knowledge graph. The triples were organized under three (mandatory) or more of the twelve total information units (i.e., *ResearchProblem*, *Approach*, *Model*, *Code*, *Dataset*, *ExperimentalSetup*, *Hyperparameters*, *Baselines*, *Results*, *Tasks*, *Experiments*, and *AblationAnalysis*). An illustration of this process is shown in Figure 1.

The difficulty of this task lies in text classification (Joulin et al., 2017) and sequence labeling (Ma and Hovy, 2016). Text classification refers to determining which of the two or more labels a one-dimensional linear sequence belongs to. Similarly, sequence labeling is used to tag each element in a one-dimensional linear sequence with a label from a set of labels. Before the popularity of deep learning, the common solutions to the sequence-labeling problem were all based on either the hidden Markov model (Zhou and Su, 2001) or conditional random field (CRF) (Ye and Ling, 2018), with CRF being the mainstream method. With the development of deep learning, convolutional neural networks (CNN) (Kim, 2014) and recurrent neural networks (RNN) (Cho et al., 2014) have achieved great success in text classification and se-

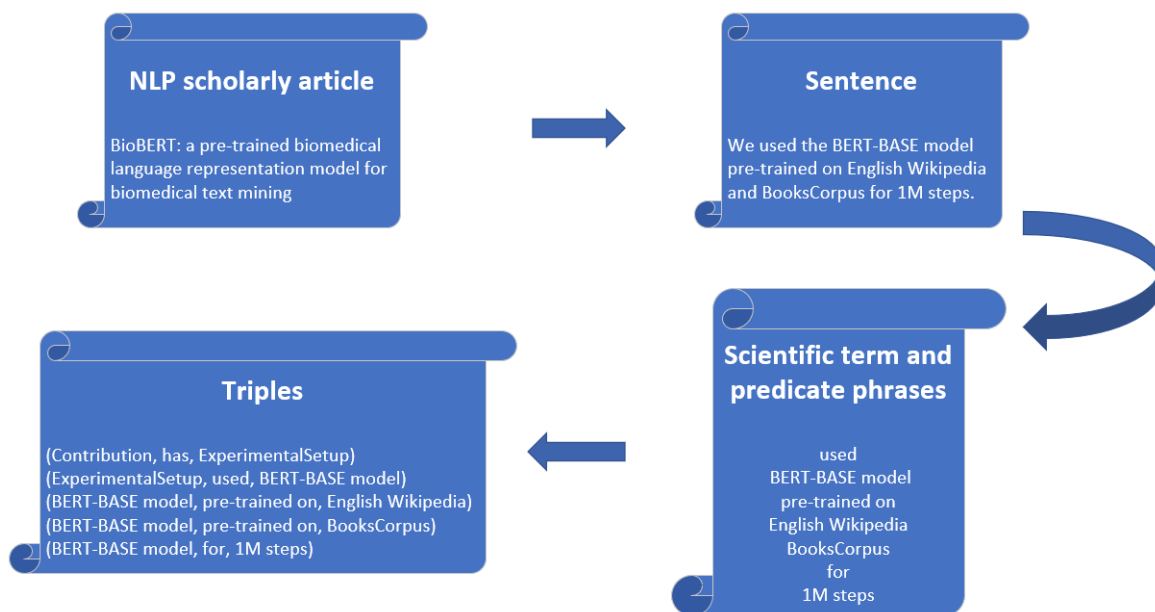


Figure 1: Example of contribution extraction of NLP scholarly articles

quence labeling. Since then, long short-term memory (LSTM) (Wang and Jiang, 2016), Bi-LSTM (Bi-directional long short-term memory), and other models (Yuan et al., 2020) have performed better than CNN and RNN in text classification and sequence labeling. However, since the introduction of bidirectional encoder representations from transformers (BERT) (Devlin et al., 2018), the accuracy and training efficiency in both text classification and sequence labeling have reached new heights.

The SemEval-2021 shared Task 11 (D’Souza et al., 2021) consists of three subtasks:

- Subtask 1: identifying contribution sentences;
- Subtask 2: identifying scientific terms and predicate phrases;
- Subtask 3: categorizing, identifying and organizing scientific terms and predicate phrases into subject-predicate-object triples.

In this study, after analysis, we converted the above three subtasks into four downstream tasks in the field of NLP: binary classification for solving Subtask 1, sequence labeling for solving Subtask 2, multiclass classification and multi-label classification for solving Subtask 3. Then, we used a pre-trained language model, BERT, to generate word embeddings and integrated them into the corresponding models for the different tasks. After

completion of the task, our results were satisfactory. Our submission ranked 7th in both Part 1 and Part 2 of Evaluation Phase 2. The implementation for our system is made available via Github<sup>1</sup>.

The remainder of this paper is organized as follows. Section 2 describes the details of the BERT model used in our system. Section 3 presents the experimental results. Finally, the conclusions are presented in Section 4.

## 2 System Description

We used a pre-trained BERT model to accomplish the task, which was defined in terms of three dataset annotation elements, where the extraction of each data element relied on the extraction of the previous data element.

### 2.1 Subtask 1: Sentence Classification

The first part of this task was to extract sentences that reflected the research contribution in the given NLP scholarly articles. We termed this sentence classification (Dao et al., 2020), where we predicted whether a sentence in an article was a contribution sentence. To this end, our approach was to pass each sentence in an article through the pre-trained BERT model to generate 768-dimensional word embeddings for each word in the sentence. The next thing we were going to do was to take the word embeddings of the first token of each sentence

<sup>1</sup><https://github.com/maxinge8698/SemEval2021-Task11>

(i.e. '[CLS]') to do sentence classification because it integrated the semantic information of the whole sentence. Then this word embeddings acquired from the previous step was connected with a fully connected layer that converted the 768-dimensional input into 2-dimensional numerical values. These values were then input into *softmax* to calculate the probability of a sentence being a contribution sentence. Finally, the probability outcomes were input into *argmax*, where, in our experimental setup, an output of 1 indicated a contribution sentence and 0 indicated the contrary. The overall architecture of the system is shown in Figure 2.

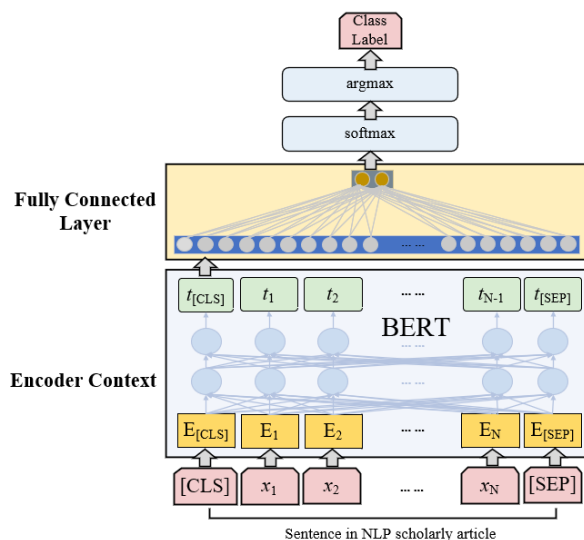


Figure 2: System of binary classification for sentence classification task

## 2.2 Subtask 2: Span Identification

Span identification (Singh et al., 2020) was a binary sequence tagging task where we classified each token in a contribution sentence to indicate whether it was part of a scientific term or predicate phrase fragment. We passed the contribution sentence identified from Subtask 1 into a pre-trained BERT model and obtained embeddings for each token in the sequence. Next, the word embeddings for each token were passed through a fully connected layer, and thereafter through *softmax* and *argmax*, where they were mapped to a class label respectively except the tokens of '[CLS]' and '[SEP]', indicating whether the token was part of a scientific term or predicate phrase fragment. The model architecture is illustrated in Figure 3. Note that the fully connected layer, *softmax*, and *argmax* were shared across all tokens.

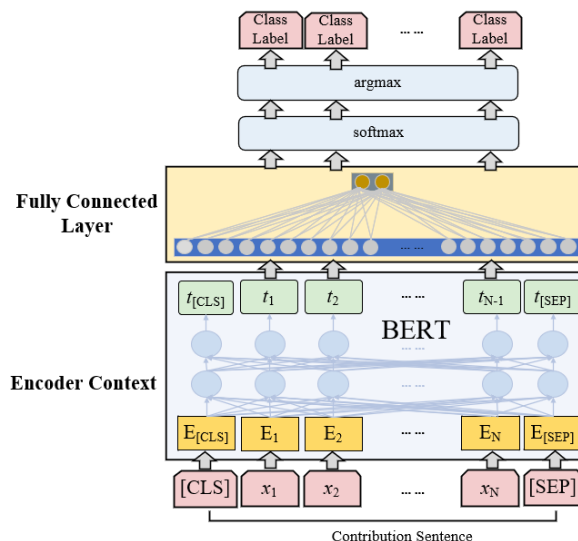
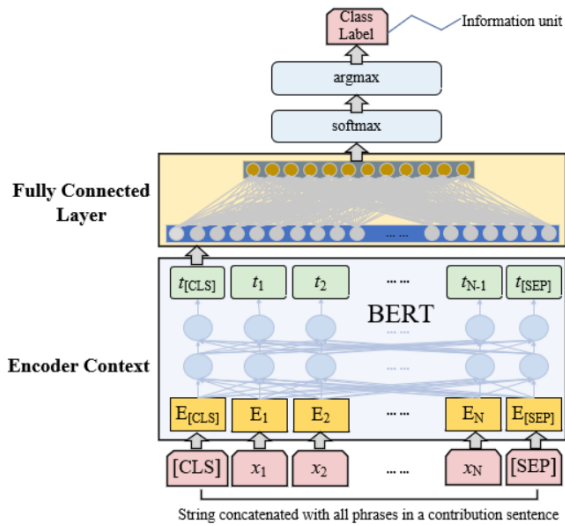


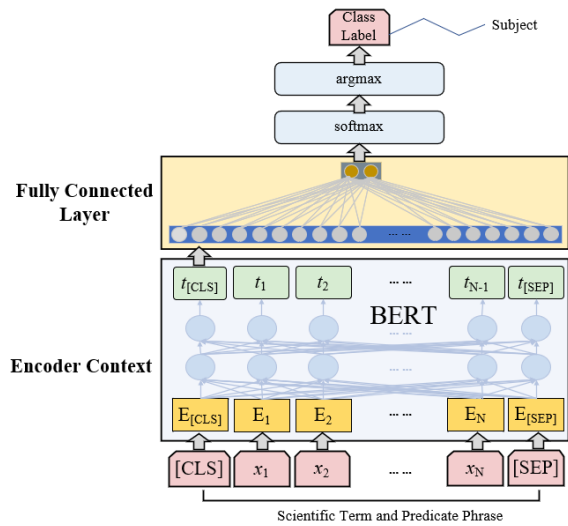
Figure 3: System of sequence labeling for span identification task

## 2.3 Subtask 3: Triple Extraction

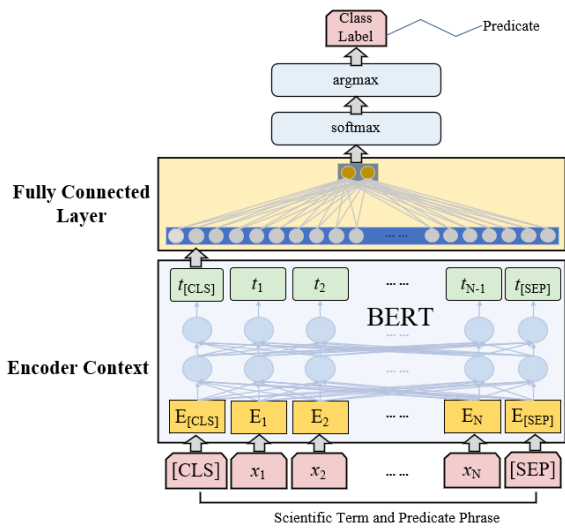
This subtask was the most cardinal and complex step in the entire task. This could be considered a relation extraction task (Lin et al., 2016), which was completed by dividing into parts information units classification and triple formation. First, it was necessary to classify all scientific term and predicate phrases in a contribution sentence extracted from Subtask 2 to determine which category of the 12 information units the extracted phrases belonged to. This was a multiclass-sequence classification problem, where we identified the unit information belonging to the scientific term and predicate phrase fragments in a given contribution sentence by concatenating all phrases into a single string to feed our model. The system architecture of this part was similar to the sentence classification system, except that there were 12 class labels and 12 output dimensions instead of 2 each (refer to Figure 4a). The next step was to identify the subject, predicate, and object in the scientific term and predicate phrases included in a contribution sentence obtained from Subtask 2 by using multi-label classification. More specifically, each scientific term and predicate phrase could be labeled with one or more of the three tags of subject, predicate and object, which could be solved by transforming the multi-label classification problem into three binary classification problems similar to the sentence classification system whereas using each phrase as input instead of each sentence. First, a binary classification system was used to



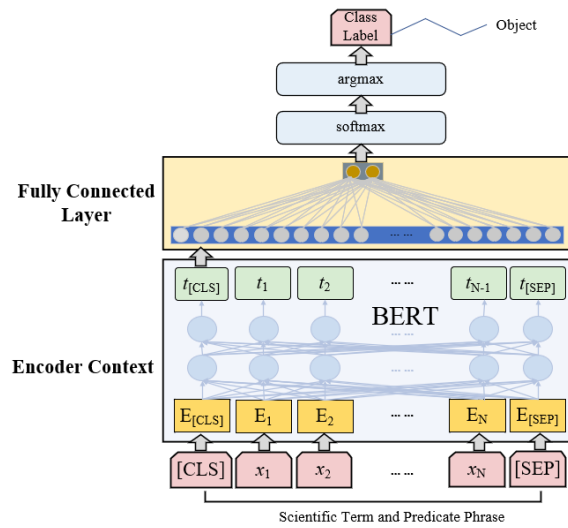
(a) Multiclass classification system for classifying information unit



(b) Multi-label classification system for identifying subjects



(c) Multi-label classification system for identifying predicates



(d) Multi-label classification system for identifying objects

Figure 4: System of multiclass classification and multi-label classification for triple extraction task

identify the subject in each scientific term and predicate phrase (refer to Figure 4b). Then, a second binary classification system was used to identify the predicate (refer to Figure 4c), and a third binary classification system was used to identify the object (refer to Figure 4d). A brief overview of this system is presented in Figure 4. At the end of the classification, for all phrases in a contribution sentence, that the corresponding label conformed to the subject-predicate-object order was found as triples iteratively from the beginning position.

### 3 Experimental Results

**Datasets.** The *NLPContributionGraph* shared task comprises a dataset of NLP scholarly articles with annotated contributions. The annotations were provided in terms of three data elements: (1) contribution sentences, (2) scientific term and predicate phrases from the sentences, and (3) (subject, predicate, object) triple statements. All the triples together formed the contribution-centered knowledge graph of the articles. The dataset released by the organizers contained 237 annotated articles as training data and 155 annotated articles as testing data for the final evaluation phases. For the training data, the annotations of each scholarly article were provided in a directory. The directory contained the full article in plain text, which was pre-processed for tokenization and sentence splitting. The annotations were provided in the following three files: (1) **sentence.txt**, specifying the annotated contribution sentence numbers from the plain text file; (2) **entities.txt**, specifying the sentence number, tab-separated from the start and end token numbers of the annotated phrase in the sentence; and (3) a directory **triples/** containing files with triples of scientific term and phrase pairs and a relation cue phrase, and the files were named to indicate the information unit that the triple data represented. For the article under the directory “training-data/natural\_language\_inference/0” as illustrated in Table 1, the sentence.txt file gave the line index of the articles contribution sentences (starting at 1). As illustrated in Table 2, the entities.txt file presented the line index which was identical to sentence.txt file, beginning position (starting at 0), end position, and corresponding text content of the scientific term and predicate phrases for each of the contribution sentences of the article. As illustrated in Table 3, the triples folder contained files named as one of the 12 information units covered in the

article, and each information unit file provided (subject, predicate, object) triples that were comprised of the scientific term and predicate phrases.

**Evaluation Metrics.** An *NLPContributionGraph* submission would be considered complete with predictions made for all three tasks (sentences, phrases, triples). The evaluation metrics that were applied are

- Sentences: *precision*, *recall* and  $F_1$ -score;
- Phrases: *precision*, *recall* and  $F_1$ -score;
- Triples: *precision*, *recall* and  $F_1$ -score overall and for each information unit.

The calculation of these three evaluation metrics is as follows:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \quad (1)$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \quad (2)$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3)$$

For the final evaluation stage of the task, the evaluation metrics is as follows:

$$F_1 = avg(F_1(Sentences), F_1(Phrases), F_1(InformationUnits), F_1(Triples)) \quad (4)$$

**Implementation Details.** The articles were split into sentences to feed into the language model, which resulted in 55,201 sentences in the training data (44,160 sentences served as the training set and 11,041 as the development set) with a maximum length of 398 words, and 33,800 sentences in testing data that originated from the evaluation phase, with a maximum length of 385 words. We used the Tensorflow framework provided by the Huggingface<sup>2</sup> library for the pre-trained BERT models and `bert-base-uncased` for binary classification, sequence labeling, multiclass classification, and multi-label classification included in this task. In addition, we fine-tuned the model using the Adam optimizer (Loshchilov and Hutter, 2018), by using a loss function of categorical cross-entropy with a learning rate of  $2 \times 10^{-5}$  and a batch size of 8 for three epochs. The activation function used by the fully connected layer was *softmax*.

<sup>2</sup><https://huggingface.co>

article_directory	sentences
training-data/natural_	2
language_inference/0	11
	13

Table 1: Part of sentences.txt corresponded to an article

article_directory	sentences	begin_offset	end_offset	text
training-data/natural_	2	30	48	Text Comprehension
	11	37	75	https://github.com/bdhingra/ga-reader
	13	43	58	machine reading

Table 2: Part of entities.txt corresponded to an article

article_directory	research-problem.txt	code.txt
training-data/natural_	(Contribution  has research	(Contribution  Code  https://
language_inference/0	problem  Text Comprehension)	github.com/bdhingra/ga-reader)
	(Contribution  has research	
	problem  machine reading)	

Table 3: Part of triples corresponded to an article

**Result and Discussion.** To allow a thorough evaluation of the systems, *NLPContributionGraph* was organized into three evaluation phases:

- **Evaluation Phase 1: End-to-end pipeline testing phase.** The participant systems were expected to output contribution sentences, their corresponding scientific terms, and predicate phrases as well as triples.
- **Evaluation Phase 2, Part 1: Phrases extraction testing.** The participant systems were given gold-annotated contribution sentences and were expected to provide purely scientific terms and predicate phrases as well as triples as extraction output.
- **Evaluation Phase 2, Part 2: Triples extraction testing.** The participant systems were given gold phrases and were expected to provide triples as the only output.

We used the Scikit-Learn<sup>3</sup> library to divide the training data into training and development sets in a 8:2 ratio. We trained our models on the training set and evaluated the prediction with the golden scores of the good performance of our approaches. For these three subtasks, the  $F_1$ -score, *Precision*, and *Recall* of our system on the development set are shown in Table 4.

<sup>3</sup><https://scikit-learn.org>

Our system achieved an average  $F_1$ -score of 0.4562 in Evaluation Phase 2, Part 1 and ranked 7th among the participating systems, and an average  $F_1$ -score of 0.6541 in Evaluation Phase 2, Part 2 and also ranked 7th among all participants. The results showed that our proposed system was effective in extracting contributions from an NLP scholarly article. The main reason was that the BERT model is a multi-layer bidirectional transformer encoder, which can be integrated into various NLP downstream tasks and achieves the best results.

Subtask	$F_1$ -score	<i>Precision</i>	<i>Recall</i>
Subtask1	0.6423	0.6554	0.6932
Subtask2	0.4768	0.5326	0.4356
Subtask3	0.4385	0.4109	0.6151

Table 4: Score of the pre-trained BERT model for the three subtasks on the development set

## 4 Conclusions

In this paper, we presented the system we submitted to the SemEval-2021 Task 11, which leveraged a pre-trained BERT model to extract contributions from an NLP scholarly article using binary classification, sequence labeling, multiclass classification, and multi-label classification. The experimental results showed that the proposed models achieved a good performance in the final evaluation phases.

Furthermore, in the three subtasks, there appeared to be significant room for improvement compared to the top-ranked participant systems. Therefore, in future research, we will attempt to generalize models with better capabilities to obtain better results.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61702443, 61966038 and 61762091.

## References

- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. [Learning phrase representations using RNN encoder-decoder for statistical machine translation](#). In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*.
- Jiaxu Dao, Jin Wang, and Xuejie Zhang. 2020. YNU-HPCC at SemEval-2020 task 11: LSTM network for detection of propaganda techniques in news articles.
- Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv*, pages 4171–4186.
- Jennifer D’Souza and Sören Auer. 2020. NLPContributions: An Annotation Scheme for Machine Reading of Scholarly Contributions in Natural Language Processing Literature. *arXiv*, pages 16–27.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 Task 11: NLPContributionGraph - Structuring Scholarly NLP Contributions for a Research Knowledge Graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Eddine Farfar, Manuel Prinz, Jennifer D’Souza, Gabor Kismihok, Markus Stocker, and Sören Auer. 2019. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. *arXiv*, pages 243–246.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. [Bag of tricks for efficient text classification](#). *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, 2:427–431.
- Yoon Kim. 2014. [Convolutional neural networks for sentence classification](#). *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1746–1751.
- Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. [Neural relation extraction with selective attention over instances](#). *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 4:2124–2133.
- Ilya Loshchilov and Frank Hutter. 2018. Fixing Weight Decay Regularization in Adam. Technical report.
- Xuezhe Ma and Eduard Hovy. 2016. [End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF](#). *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 2:1064–1074.
- Paramansh Singh, Siraj Sandhu, Subham Kumar, and Ashutosh Modi. 2020. [newsSweeper at SemEval-2020 task 11: Context-aware rich feature representations for propaganda classification](#). *arXiv*.
- Shuohang Wang and Jing Jiang. 2016. [Learning natural language inference with LSTM](#). *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 1442–1451.
- Zhi Xiu Ye and Zhen Hua Ling. 2018. Hybrid semi-markov CRF for neural sequence labeling. *arXiv*, pages 235–240.
- Li Yuan, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2020. [Graph Attention Network with Memory Fusion for Aspect-level Sentiment Analysis](#). *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 27–36.
- GuoDong Zhou and Jian Su. 2001. [Named entity recognition using an HMM-based chunk tagger](#). (July):473.



# ITNLP at SemEval-2021 Task 11: Boosting BERT with Sampling and Adversarial Training for Knowledge Extraction

Genyu Zhang, Yu Su, Changhong He, Lei Lin, Chengjie Sun, Lili Shan

Intelligence Technology and Natural Language Processing Lab,

School of Computer Science and Technology,

Harbin Institute of Technology

{gyzhang, suyu, changhong.he}@insun.hit.edu.cn

{cjsun, linl, shanll}@insun.hit.edu.cn

## Abstract

This paper describes the winning system in the End-to-end Pipeline phase for the NLP-ContributionGraph task. The system is composed of three BERT-based models and the three models are used to extract sentences, phrases and triples respectively. Experiments show that sampling and adversarial training can greatly boost the system. In End-to-end Pipeline phase, our system got an average F1 of 0.4703, significantly higher than the second-placed system which got an average F1 of 0.3828.

## 1 Introduction

The Knowledge Graph (KG) describes the concepts, entities and their relationships in the objective world in a structured form, expresses Internet information in a form closer to the human cognitive world. Information extraction is the first step of the KG construction. Information extraction is a technology that extracts structured information such as entities and relationships from semi-structured or unstructured data automatically. Similarly, as the rate of research publications increases, it is critical to construct Knowledge Graphs to represent scholarly knowledge efficiently. The target of the NLPContributionGraph task (D'Souza et al., 2021) is to find a systematic set of patterns of subject-predicate-object statements for the semantic structuring of scholarly contributions that are generically applicable for NLP research articles, then apply the discovered patterns in the creation of a larger annotated dataset for ingesting the dataset into the Open Research Knowledge Graph infrastructure to assist users manually manage their article contributions. Our task consists of three sub-tasks: Sentences Extraction (SE), Phrases Extraction (PE) and Triples Extraction (TE).

The dataset used in the NLPContributionGraph task contains hundreds of Natural Language Pro-

cessing (NLP) scholarly articles annotated for their contributions. Each article is written in English and contains three types of annotation information: 1) contribution sentences; 2) scientific term and predicate phrases from the sentences; and 3) subject-predicate-object triple statements from the phrases toward KG building.

Our code is available at <https://github.com/itnlp606/nlpcb-graph>.

## 2 Related Work

In recent years, pretrained language models (Peters et al., 2018; Devlin et al., 2019; Sun et al., 2019; Lan et al., 2020) have achieved impressive performance in various NLP tasks including information extraction. BERT (Devlin et al., 2019) uses Bidirectional Transformers (Vaswani et al., 2017) to pretrain the model on the Masked Language Model (MLM) task and the Next Sentence Prediction (NSP) task and advances the state-of-the-art for eleven NLP tasks. The system presented in this paper is based on fine-tuning BERT. This section will introduce two strategies to boost the BERT model.

### 2.1 Sampling

In classification tasks, we often encounter uneven distribution of positive and negative samples. Under such distribution, the model may not be able to make accurate predictions. The trained model naturally tends to predict the majority set, and the minority set may be considered as noise. Compared with the majority set, the minority set is more likely to be misclassified. Modifying loss function (Lin et al., 2017; Li et al., 2019) and sampling methods (Chawla et al., 2002; Liu et al., 2009) are valid approaches to solve this problem, and the later was adopted in our system. Oversampling achieves sample balance by increasing the number of minority samples in classification.

The most direct method is to simply copy the minority samples to form multiple records. The disadvantage of this method is that if the sample features are few, the model is easy to overfit. SMOTE (Chawla et al., 2002) interpolates between samples of the minority class to generate additional samples. Under-sampling achieves sample balance by reducing the number of samples of the majority class in classification. The most direct method is to randomly remove some samples of the majority class. EasyEnsemble (Liu et al., 2009) divides the majority samples into several parts randomly, so the data of each part is equal to the number of minority samples. Then, multiple models are trained on different parts of data, and the output of each model will be integrated. BalanceCascade (Liu et al., 2009) combines a subset of the majority class with the minority class to train the model, then discards the samples that are correctly classified in the next round, so that the subsequent base learner can pay more attention to those samples that are incorrectly classified. Our model uses under-sampling for sentences extraction and triples extraction. For different tasks, diverse sampling strategies have been adopted.

## 2.2 Adversarial training

As machine learning model is vulnerable to some small worst-case perturbations, adversarial training (Goodfellow et al., 2014) aims to make the AI systems safer by improving the robustness of the model. In Computer Vision tasks, adversarial training usually hurts the generalization of the model. However, Miyato et al. (2017) adopted adversarial training in text classifying by applying perturbations to the word embeddings, which can improve both generalization and robustness of the NLP models.

Considerable efforts have been made to find better adversarial perturbations. The Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2014) generates adversarial examples by formulation:

$$\hat{\mathbf{x}} = \mathbf{x} + \mathbf{r}_{adv}$$

$$\mathbf{r}_{adv} = \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y))$$

where  $\mathbf{x}$  are the embeddings of the input text,  $\mathbf{r}_{adv}$  are the adversarial perturbations,  $\boldsymbol{\theta}$  are model parameters,  $\hat{\mathbf{x}}$  are embeddings of adversarial examples that are used to update the model. The Fast Gradient Method (FGM) (Miyato et al., 2017) is another generation of FGSM in which the pertur-

bations are normalized by gradients:

$$\mathbf{r}_{adv} = -\epsilon \frac{\mathbf{g}}{\|\mathbf{g}\|_2}$$

where  $\mathbf{g} = \nabla_{\mathbf{x}} \log p(y | \mathbf{x}; \hat{\boldsymbol{\theta}})$ .

Madry et al. (2018) used a min-max formulation as follows to cast both attacks and defenses into a common theoretical framework,

$$\min_{\theta} \left\{ \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \max_{r \in \mathcal{S}} L(\theta, x + r, y) \right] \right\}$$

in this formulation, the inner maximization problem describes attack which aims to find the most adversarial data leading to a high loss, the outer minimization problem describes defense which aims to find the most robust model. They also proposed Projected Gradient Descent (PGD) that uses an iterative algorithm to generate the most adversarial data.

The Friendly Adversarial Training (FAT) (Jingfeng et al., 2020) adopted by our team is an early-stopped version of PGD, its adversarial data was generated by a min-min formulation as following:

$$\tilde{x}_i = \arg \min_{\tilde{x} \in B(x_i)} \ell(f(\tilde{x}), y_i)$$

$$\text{s.t. } \ell(f(\tilde{x}), y_i) - \min_{y \in \mathcal{Y}} \ell(f(\tilde{x}), y) \geq \rho$$

different from PGD, FAT generates friendly adversarial data rather than the most adversarial data,  $\rho > 0$  is a margin that indicates the confidence of adversarial data being misclassified. FAT is more computationally efficient than PGD, and model trained with FAT can reach higher accuracy.

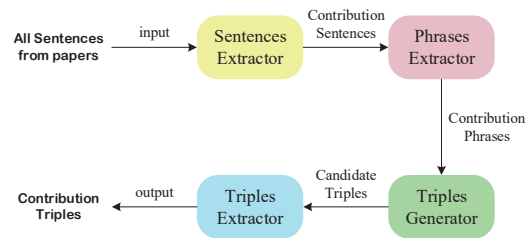


Figure 1: System Overview

## 3 System Description

For three sub-tasks in NLPContributionGraph, we designed four modules to implement these tasks. These modules use fine-tuning BERT with FAT as

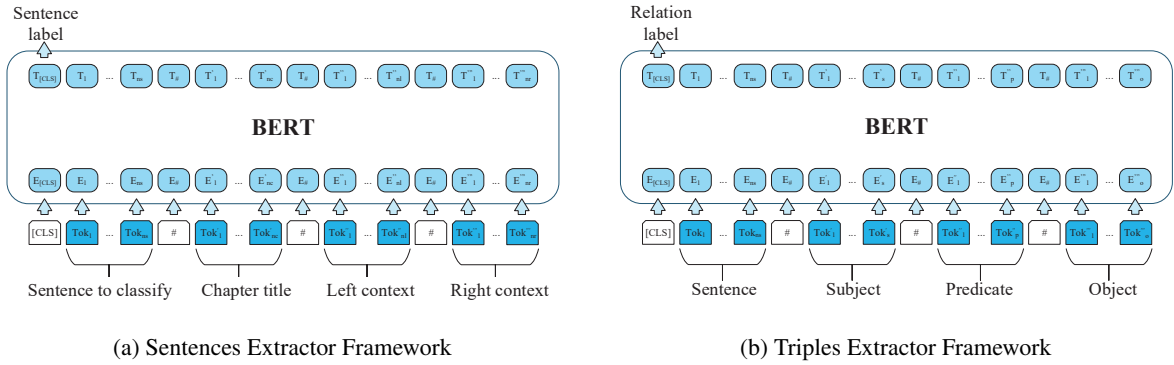


Figure 2: Both of the Sentences Extractor and the Triples Extractor are build by the BERT-based classification model.

the base model and adopt different boosting methods to improve overall results. The full framework of the system is shown in Figure 1. The three extractors can extract contribution information by classification. The Triples Generator can convert discrete phrases into triples. These modules will be explained further in following sections.

### 3.1 Sentences Extractor

The system uses the Sentence Extractor to solve the SE task. The extractor can extract the sentences that have the contribution information such as research problem, code, etc. As shown in Figure 2a, the Sentences Extractor uses the sentence context and paragraph heading as additional features and uses BERT as a binary classifier to determine whether the sentence contains the contribution information. In an annotated paper, most sentences do not contain the contribution information. Therefore, we adopted an under-sampling strategy. In the training process, the ratio of positive samples and negative samples is fixed to an integer for each batch to ensure that the model will not overfit on negative samples. The ratio is a hyperparameter that needs to be tuned in the training process.

### 3.2 Phrases Extractor

The Phrases Extractor can extract contribution phrases from the sentences to solve the PE task. For this task, the BERT-based sequence labeling model is effective. The phrases predicted by the trained model will sometimes be incomplete, resulting in high recall and low precision. Fortunately, ensembling learning can solve this problem well. In the competition, we trained ten different models by ten-fold cross-validation. After training, the

trained models will make their own predictions, and the module will count the number of votes for each phrase. Only phrases with more than a certain number of votes will be seen as a valid output.

### 3.3 Triples Generator

After the Phrases Extractor completes the prediction, we can obtain discrete phrases. The role of the Triples Generator is to convert these phrases into triples through permutation and combination. This section will introduce two methods to finish this task.

**Language Model Approach** Language models are usually used to evaluate the probability of a sentence. The triples to be extracted are composed of subject, predicate and object, which are components of a sentence. This approach uses a language model to evaluate the probability of triples. The input of the module is all permutations of contribution phrases, and the permutation that has the highest probability will be the output of the module, which are candidate triples.

**Combination Approach** Due to the lack of data, the prediction made by language model is not accurate. In the annotated data, the order of about ninety percent of the triples is sequential. In order to deal with the insufficient representation ability of the language model, we directly use the combination of all serial phrases as the output of this module. In the competition, we adopted Combination Approach as the Triples Generator.

### 3.4 Triples Extractor

The Triples Extractor can classify all candidate triples based on the BERT model. As shown in Figure 2b, sentences and triples are separated by

hash marks, and inputted into BERT for classification. Unfortunately, the trained model will easily overfit on negative samples due to the large number of combinations. Therefore, we need to adopt the under-sampling strategy to boost the base BERT. For complex combinations, the module combines two strategies to select negative samples:

**Random Replacement (RR)** For each of the three phrases in the positive sample, we will randomly select one and replace it with another phrase.

**Random Selection (RS)** Randomly select three phrases that are not positive samples.

The module combines the two sampling methods above to generate negative samples. For each batch, it fixes the ratio of positive and negative samples to generalize better.

## 4 Experimental Setup

In this section, we did some ablation analysis on the validation set for the boosting methods proposed in this paper, and gave some analysis through the experimental results.

### 4.1 Adversarial Training

We tested the performance of different adversarial training approaches. Table 1 shows that FAT achieved the best results in all three tasks. Especially in the SE task, adversarial training can greatly improve the model’s performance. During the experiment, we also found that if adversarial training is not applied, training will converge in an average of five epochs. If the system uses adversarial training, the training will last for about twenty epochs and will continuously improve the performance on the validation set. The perturbations added by the adversarial training make the model generalize better. In addition, ensembling is not applied in the PE task, so the F1 score of this task is low. This issue would not affect the experimental results.

Task	Natural	FGM	FAT
SE	0.4112	0.5527	<b>0.5615</b>
PE	0.2011	0.2128	<b>0.2231</b>
TE	0.4641	0.4740	<b>0.5176</b>

Table 1: F1 scores of Natural training (no adversarial training), FGM and FAT.

### 4.2 Sentences Features

We randomly selected five domains of papers to test the effect of different features on the SE task. These domains are Question Answering (QA), Relation Extraction (RE), Sentence Classification (SC<sub>1</sub>), Sentence Compression (SC<sub>2</sub>) and Text Generation (TG).

Table 2 shows that adding either context or title can significantly improve the accuracy of classification and the best results can be achieved by concatenating both of them.

Domain	Natural	Title	Context	T&C
QA	0.4068	0.5294	0.6471	<b>0.6977</b>
RE	0.4516	0.5128	0.5781	<b>0.5827</b>
SC <sub>1</sub>	0.3636	0.5417	0.7391	<b>0.7826</b>
SC <sub>2</sub>	0.5600	0.5714	0.5926	<b>0.6667</b>
TG	0.4681	0.5424	0.5385	<b>0.5763</b>

Table 2: F1 scores while adding different features. T&C means adding both Title and Context.

### 4.3 Triples Extractor Sampler

In the TE task, RR and RS are applied as the sampling methods. Without the under-sampling strategy, the model is difficult to converge. Table 3 shows the performance of different sampling methods on papers in various domains. Among them, the combination of RR and RS strategies achieved the best results. The diversity of the sampling strategies improves the generalization.

Domain	RR	RS	RR&RS
QA	0.3621	0.3592	<b>0.4267</b>
RE	0.3782	0.4033	<b>0.4163</b>
SC <sub>1</sub>	0.3359	0.3505	<b>0.3692</b>
SC <sub>2</sub>	0.4585	0.4623	<b>0.4777</b>
TG	0.4900	0.5351	<b>0.5748</b>

Table 3: Macro-F1 scores with different triple sampling methods

### 4.4 Evaluation Results

In End-to-end Pipeline phase, our system got F1 scores of 0.5619, 0.4522 and 0.1379 in tasks SE, PE, TE, respectively. Our system has achieved good results on tasks SE and PE, but task TE can still be improved. Since our system only considers sequential triples, some triples will be missed, which can be a defect of our system.

## 5 Conclusion

BERT is a powerful model that has considerable applications in numerous fields of NLP. Using BERT in the NLPContributionGraph task allows researchers to read papers more efficiently. The methods of adversarial training and sampling proposed in this paper can greatly boost the performance of BERT on this task and can also offer some thoughts for future work on knowledge extraction of papers.

## Acknowledgements

This work was supported by the National Key R&D Program of China via grant 2018YFC0830700, National Natural Science Foundation of China (NSFC) via grant 61772156.

## References

- V. Nitesh Chawla, W. Kevin Bowyer, O. Lawrence Hall, and Philip W. Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, pages 321–357.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: Nlpcontributiongraph - structuring scholarly nlp contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- J. Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *international conference on learning representations*.
- Zhang Jingfeng, Xu Xilie, Han Bo, Niu Gang, Cui Lizhen, Sugiyama Masashi, and Kankanhalli Mohan. 2020. Attacks which do not kill training make adversarial learning stronger. *ICML*, pages 11278–11287.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. *ICLR*.
- Buyu Li, Yu Liu, and Xiaogang Wang. 2019. Gradient harmonized single-stage detector. *national conference on artificial intelligence*.
- Tsung-Yi Lin, Priya Goyal, B. Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. *ICCV*, pages 318–327.
- Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. 2009. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, pages 539–550.
- Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. *international conference on learning representations*.
- Takeru Miyato, M. Andrew Dai, and J. Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. *international conference on learning representations*.
- E. Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and S. Luke Zettlemoyer. 2018. Deep contextualized word representations. *north american chapter of the association for computational linguistics*.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv: Computation and Language*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, N. Aidan Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 30 (NIPS 2017)*, pages 5998–6008.

## Appendix

This section will list the hyperparameters used in the competition, which can help researchers replicate the experiments conducted in this paper.

- **Global**
  - Batch size: 16
  - Learning rate (Adam): 5e-5
  - Pretrained model: BERT-base
  - Word embedding size: 512
  - Hidden layer size: 768
- **Task SE**
  - Number of sentences in the context: 2
  - Positive and negative sample ratio: 1:3
- **Task TE**
  - Positive and RS sample ratio: 1:3
  - Positive and RR sample ratio: 1:3
  - Positive and negative sample ratio: 1:6

# Duluth at SemEval-2021 Task 11: Applying DeBERTa to Contributing Sentence Selection and Dependency Parsing for Entity Extraction

Anna Martin & Ted Pedersen

Department of Computer Science

University of Minnesota

Duluth, MN 55812, USA

{mart5877, tpederse}@d.umn.edu

## Abstract

This paper describes the Duluth system that participated in SemEval-2021 Task 11, NLP Contribution Graph. It details the extraction of contribution sentences and scientific entities and their relations from scholarly articles in the domain of Natural Language Processing. Our solution uses deBERTa for multi-class sentence classification to extract the contributing sentences and their type, and dependency parsing to extract phrases from each sentence and format into subject-predicate-object triples. Our system ranked fifth of seven for Phase 1: end-to-end pipeline, sixth of eight for Phase 2 Part 1: phrases and triples, and fifth of eight for Phase 2 Part 2: triples extraction.

## 1 Introduction

The rapid rate at which scientific literature grows makes it difficult to keep up with new research even in one’s own field. Automated solutions are challenging since text formatted and written for human consumption does not lend itself to machine processing.

(Jaradeh et al., 2019) have proposed a knowledge graph based system for collecting and structuring articles in a machine-readable format. This requires the annotation of many scholarly articles, a task that is time consuming to do by hand. The purpose of this SemEval task (D’Souza et al., 2021) is to enable the construction of a scholarly contributions graph over English language NLP articles by automating the task of annotating scientific papers.

This annotation processes consists of:

1. selecting sentences that describe the contribution of the article and outputting them to a sentences.txt file,
2. extracting scientific entities and relations from the selected sentences and outputting them to an entities.txt file,

3. and structuring the entities and relations into triples of the form subject-predicate-object. These triples are sorted into one of twelve information units, which are labels that describe the type of contribution being made by the sentence outlined in the triples file.

Following is an example of the annotation process using a sentence taken from the training data with the human-labeled entities in braces:

We [apply] [dropout] of [0.4] [to layers], [0.3] [to RNN layers], [0.4] [to input embedding layers], [0.05] [to embedding layers], and [weight dropout] of [0.5] [to the RNN hidden-to-hidden matrix].

This task requires the 1) identification of this sentence as a contribution sentence belonging to the information unit HYPERPARAMETERS, 2) the extraction of the entities in braces, and 3) the formatting of those entities into triples. This example and various others will be used throughout this paper to help contextualize our system description.

SemEval-2021 Task 11 was organized into three phases, called Phase 1: end-to-end pipeline, Phase 2 Part 1: phrases and triples, and Phase 2 Part 2: triples extraction. Phase 1 tested the entire system, scoring for sentence extraction, phrases extraction, and triples extraction. The gold sentences.txt files were released for use in Phase 2 Part 1, in order to test phrases and triples extraction given perfect sentence selection. The gold entities.txt files were released for use in Phase 2 Part 2, in order to test triples extraction given perfect phrases selection.

There are four datasets mentioned in this paper. Gold data refers to the sentences.txt, entities.txt, and triples folders released by the organizers after each phase of the task. Test data refers to the data used to test the system during evaluation phases. Training data refers to the entire training dataset provided by the organizers. The trial dataset is the

segment of the training dataset that we used to test the Duluth system during validation, before the first evaluation phase began. For more information on how this dataset was created, see Appendix B.

Our approach<sup>1</sup> employs a variety of techniques to address each component of the task. The selection of contribution sentences was done by fine-tuning the base deBERTa model (He et al., 2021) on the training data, as fine-tuned BERT (Devlin et al., 2019) models have been found to perform well on multi-class text classification tasks (Liu and Wang-perawong, 2019). The fine-tuned model is used to classify each sentence as either non-contributing, or one of the twelve information units.

The selected contribution sentences were then tagged to indicate likely scientific entities using a Maximum Entropy Markov Model (MEMM) (Bird et al., 2009) that was trained on the noun phrases selected as entities in the training data.

A dependency parse (Manning et al., 2014) was then found for each sentence. The dependency parse and entity tags were then leveraged to select contributing phrase spans and triples by using the entity tags to determine whether a subject or object noun phrase ought to be considered, and using the dependencies to extract Subject–Predicate–Object patterns from sentence.

## 2 Previous Work

Two previous SemEval tasks were also concerned with the extraction of relations and key phrases from scientific publications: SemEval 2017 Task 10: (ScienceIE - Extracting Keyphrases and Relations from Scientific Publications) (Augenstein et al., 2017), and SemEval 2018 Task 7: (Semantic Relation Extraction and Classification in Scientific Papers) (Gábor et al., 2018).

Many of the approaches in these tasks use neural models to extract entities and their relations. The AI2 system (Ammar et al., 2017) at SemEval-2017 Task 10, which ranked first and second for task scenarios one and three respectively, approached this by building separate entity and relation models, each of which contain layers of LSTMs. The ETH-DS3Lab system (Rotsztein et al., 2018) at SemEval-2018 Task 7, which ranked first in three of four subtasks, built an entity and relation classifier using a combination of RNNs and CNNs.

Other approaches used supervised machine

---

<sup>1</sup>Code is available at <https://github.com/amartin94/DuluthSemEval2021Task11>.

learning algorithms while leveraging grammatical features. The LIPN system (Hernandez et al., 2017) approached SemEval-2017 Task 10 by first filtering possible keyphrases by labeling phrases with their POS sequence. Candidate keyphrases are filtered by comparing the POS tags of the phrase with POS sequences developed from the training data. They then trained a CRF model using the candidate phrases labeled with IOB tags. They were able to improve recall for keyphrase extraction by filtering candidate sentences before using a CRF.

## 3 Selection of Contribution Sentences

We approached the selection of a contribution sentence as a multi-class sentence classification problem with 13 classes, where each of the twelve information units is a class, and class 0 represents non-contributing sentences. Although the subtask of sentence extraction could be performed using a binary classifier to label sentences as either contributing or non-contributing, we decided to sort the sentences further into their information units. The alternative would require classifying phrases or triples further down the pipeline; the benefit to classifying the contributions during the sentence extraction step is that the whole context of each sentence is taken into account.

The main challenge with this approach was in the unbalanced nature of the data; 90.11% of the sentences used to train the classification model for Phase 1: end-to-end pipeline were non-contributing sentences, and the standard deviation of the frequencies of contributing classes was 8.59%. These frequencies can be seen in Appendix D.

Logistic regression and decision tree classifiers (Pedregosa et al., 2011) were not able to identify the underrepresented classes such as TASKS and DATASET, and were heavily skewed towards the dominant class of non-contributing sentences. During initial experiments using the trial dataset described in Appendix B, decision tree classifier earned a macro-F1 score of 0.1736 and the logistic regression classifier earned a macro-F1 score of 0.1738. The base deBERTa model performed better than both decision tree and logistic regression classifiers on the trial dataset, resulting in a macro-F1 score of 0.3079.

### 3.1 Phase 1: Classifying Sentences using DeBERTa

For Phase 1: end-to-end pipeline testing, we fine-tuned the base deBERTa model on sentences from the training dataset to create a thirteen-class sentence classification model, using HuggingFace transformers’ deBERTa for Sequence Classification model (Wolf et al., 2020). Hyperparameter settings can be found in Appendix E.

The provided training dataset includes sentences files that contain a list of the indexes of contributing sentences for each scholarly article. It also contains files for each information unit provided in json format; each of these files include the full sentences belonging to its specified information unit. We labeled the contributing sentences with their information units by looking up each sentence in the information unit json files. Sentences from the articles in the training dataset that were not included in the sentences files were labeled as non contributing.

### 3.2 Phase 2: Classifying Given Sentences

During evaluation Phase 2 the gold contribution sentences for the test data were given to all participants by the task organizers in sentences files containing the indexes of contributing sentences. Given this, we altered the sentence classification step by fine tuning deBERTa only on contribution sentences from the training data. This resulted in a twelve-class classifier that labeled the test data sentences according to their predicted information unit. The reason why a classification step was still required here is that the triples extraction task further down the pipeline require the sentences to be classified according to their information unit. Without the information unit json files, the information unit labels must be predicted.

Observing that the sentence indexes in the given sentences files appeared to be sorted by information unit, we adjusted the output from the sentence classifier so that chunks of consecutive sentence indexes would all receive the same label. This was performed by searching the classifier output for spans of consecutive sentences where the classifier vacillated between two commonly confused information units, such as EXPERIMENTAL SETUP and HYPERPARAMETERS. For each of these spans, the information unit that was more frequent within the span would be assigned to every sentence.

## 4 Entity and Relation Extraction

The Duluth system for Phase 1: end-to-end pipeline combined statistical and rule-based approaches for extracting scientific entities and their predicates from the contribution sentences. We used a dependency parser to extract noun phrases and their predicates, and trained an maximum-entropy Markov model (MEMM) on the training data entities files to predict whether each noun phrase contains a scientific entity.

### 4.1 MEMM Entity Extraction

For Phase 1, in order to tag likely scientific entities in the test data, we trained a MEMM on the provided training data. The features we used include:

- current word type,
- current part-of-speech tag,
- current word shape,
- current IOB tag (I or B if present in the scientific entities list, O if not), and
- the above features for the previous word.

We generated the scientific phrases list from the training data entities files, by extracting the noun phrases from the phrase spans and inputting them into a file to be looked up by the MEMM entity extractor.

For Phase 2 Part 1: phrases and triples, the model was altered to only perform IO tagging. This change was made to address the fact that sometimes individual words appear in different positions in different phrases. For example, the noun “loss” appears in 41 different phrases in the scientific phrases list, sometimes in the beginning of a phrase as in “loss function”, and sometimes in the end of a phrase as in “cross entropy loss”.

We used these features to train NLTK’s Maxent-Classifier method (Bird et al., 2009) with maximum iterations set to 40. The model achieved a testing accuracy of 0.995. We used the Viterbi algorithm to derive the most likely IO tags for every word in each sentence.

The model was able to identify some scientific entities in the test data that aren’t present in the scientific entities list derived from the training data. However, a complicating factor is that not all entities that must be found can be considered to be exclusively scientific entities. For example, terms like *ReLU*, and *SCIBERT* are clearly specific to



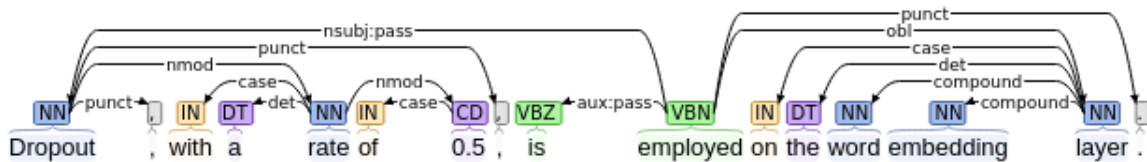


Figure 1: This is a dependency parse of example sentence from corenlp.run/, which shows the information provided to the phrase extraction system by the dependency parser. The dependencies are traced by the Duluth system in order to extract subject-predicate-object phrases from each sentence.

natural language processing and machine learning, but words such as *image*, *action*, *humans*, and other common nouns were also present in the entities training data. Since there is a wide range of specificity in the terms that must be extracted as entities, filtering the sentences through the MEMM entity tagger ultimately worsened the system’s performance in Phase 2 Part 1: phrases and triples from a total F1 score of .4634 to .4299. This is because it filtered out sentences which were contribution sentences but whose subject phrases did not contain nouns tagged as scientific entities by the classifier.

## 4.2 Dependency Parsing

We used Stanford Core NLP’s dependency parser (Manning et al., 2014; Chen and Manning, 2014) to generate a dependency parse for each contribution sentence. We used the dependency parse of each sentence to extract the root verb phrase from each sentence, its noun subject phrase, and dependent object phrases. In Phase 1, if neither the subject of the sentence nor the words dependent on it within its noun phrase were tagged as scientific entities, then the sentence would be ignored. Any object noun phrases not containing a scientific entity were also ignored. The intention was to create an outline of each contribution sentence that included only the relevant noun phrases and the relation between them.

Figure 1 illustrates the dependency parse of our running example. The system would extract the verb phrase “is employed” as the relation by finding the ROOT of the sentence (“employed”), finding the adjacent dependency (“is”), and concatenating the two into the phrase “is employed”. Next, it would extract the dependent nsubj “Dropout” as the subject entity by searching for nsubj types dependent on “employed”. Lastly, the system would extract the noun phrase which is dependent on the verb phrase “is employed” as the object entity. This would be accomplished by finding the noun

dependent on “employed”, which is “layer”, and building the noun phrase “word embedding layer” from the words dependent on “layer”. The phrases “Dropout” and “word embedding layer” would be labeled as scientific phrases, which means that this subject-predicate-object phrase would be kept.

In Phase 2, we altered our system so that it would not throw away any sentences, since it was provided with the gold contribution sentences made available by the task organizers. Rather, if the subject was a pronoun and the subject phrase did not contain a scientific entity, then the subject phrase would be removed. If there was a previously selected noun phrase from the same information unit, that phrase would replace the removed subject phrase. Otherwise, the name of the information unit would be used instead. The intention was to handle cases where a pronoun referring to an entity from the previous sentence was the subject of the verb phrase.

## 5 Triples Extraction

During evaluation Phase 1: end-to-end pipeline, almost all of the task of extracting Subject–Predicate–Object triples into information units files was already performed by previous steps. The entity extractor described in section 4 extracts phrase spans three at a time, following the subject-predicate-object format needed to organize phrase spans into triples. The sentence extractor described in section 3 classifies sentences into their information units, so the class label for the sentence that the triple is extracted from can be used to determine the information unit that the triple belongs to.

For Phase 1: end-to-end pipeline and Phase 2 Part 1: phrases and triples, the system formed triples based on the subject, predicate, and object phrases determined by the entity selection process described in section 4.

For example, if the sentence “Dropout, with a rate of 0.5, is employed on the word embedding layer” is classified by the sentence extractor as

belonging to the information unit EXPERIMENTAL SETUP, and the phrases “Dropout”, “is employed”, and “word embedding layer” were selected as a subject-predicate-object pattern, then triples would be formatted in the experimental-setup.txt triples file like so:

(Contribution  has  Experimental Setup)
(Experimental Setup  has  Dropout)
(Dropout  is employed   word embedding layer)

## 5.1 Triples Formatting

While most of the information units’ triples were formatted in the training data following the pattern in section 5, the information units Code and Research Problem were formatted following these patterns, respectively:

(Contribution  Code  url)
(Contribution  has research problem  phrase)

The triples files for information units Code and Research Problem were handled separately from the others in order to reflect these differences.

## 5.2 Phase 2 Part 2: Adapting to the Released Entities Files

The main problem with the method for building triples described in section 5 is that it does not address the fact that triples often build on each other and overlap. Triples build on each other in two ways: the first item of a triple may be the last item in the previous triple; and the first item of a triple may be the first item in a previous triple.

The Duluth system solution for Phase 2 Part 2: triples extraction attempts to imitate these patterns by following these rules: while items in the entities file alternate between noun phrases and predicates, then triples are formed where the middle item for each triple is a predicate phrase and the phrases on either side of it are the noun phrases in the entities file on either side of the predicate in the entities file. This creates the first pattern identified above. However, if two consecutive noun phrases are found in the entities file, then the second noun phrase is paired with the previous subject-predicate pair, rather than the previous noun phrase. If there is no previous subject-predicate pair, then the second

Information Unit	F1
None	.9494
Ablation Analysis	.1516
Approach	.0000
Baselines	.2559
Code	.7857
Dataset	.0000
Experimental Setup	.2466
Experiments	.0000
Hyperparameters	.2358
Model	.1778
Research Problem	.4192
Results	.3165
Tasks	N/A

Table 1: F1 scores for each information unit based on evaluation on test data. Macro-F1 score = .2949. Weighted-F1 score = .8866.

noun phrase is paired with the name of the information unit and the predicate *has*. This creates the second pattern identified above.

## 6 Sentence Selection Results

The official competition F1 score on the gold standard test data for selection of contribution sentences in Phase 1: end-to-end pipeline was 0.38095. This score evaluates the results as a binary classification problem, where sentences are either contributing or non-contributing. Because our sentence selector performs multi-class classification, labeling each sentence as either non-contributing or belonging to one of twelve information units, we provide a confusion matrix in Table 2 that shows the detailed results by information unit, as well as a break down of individual F1 scores for each information unit in Table 1. The official competition F1 score for information units in Phase 1 was 0.6441.

### 6.1 Confusion Matrix Analysis

The high frequency of false positives and false negatives for the non-contribution class (None in Table 2) is likely due to its high frequency, as 90.11% of the sentences from the training dataset belong to this class. This shows that using a BERT model without any filtering or sampling techniques is not sufficient to accurately handle the unbalanced nature of this dataset. We will focus our discussion here on the most frequently confused information units.

30.19% (109 of 361) of sentences describing

Predicted Class	Gold Class												
	N	AA	A	B	C	D	ES	E	H	M	RP	R	T
None	27,922	113	123	55	7	10	178	223	59	446	230	405	0
Ablation Analysis	72	21	0	0	0	0	0	8	0	0	0	<b>14</b>	0
Approach	0	0	0	0	0	0	0	0	0	0	0	0	0
Baselines	146	0	0	38	1	0	1	5	2	5	2	2	0
Code	10	0	0	0	33	0	0	0	0	0	0	0	0
Dataset	0	0	0	0	0	0	0	0	0	0	0	0	0
Experimental Setup	125	0	0	0	0	0	73	5	<b>27</b>	1	0	0	0
Experiments	0	0	0	0	0	0	0	0	0	0	0	0	0
Hyperparameters	136	0	0	0	0	0	<b>109</b>	4	52	0	0	0	0
Model	222	0	<b>7</b>	2	0	0	0	2	0	75	2	3	0
Research Problem	88	0	0	0	0	0	0	1	0	3	118	0	0
Results	327	<b>28</b>	0	0	0	0	0	<b>87</b>	0	1	1	201	0
Tasks	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: Confusion matrix for thirteen-class sentence classification. The quantities in bold face correspond with the boldfaced quantities in section 6.1.

EXPERIMENTAL SETUP were falsely labeled as belonging to the information unit HYPERPARAMETERS by our classifier, and 19.28% (**27** of 140) of sentences describing HYPERPARAMETERS were falsely labeled as EXPERIMENTAL SETUP. This makes sense, as sentences describing experimental setup may include discussion of hyperparameters.

Sentences belonging to the information unit ABLATION ANALYSIS were incorrectly labeled as RESULTS 17.28% (**28** of 162) of the time (the classification of RESULTS was more successful, only being incorrectly classified as ABLATION ANALYSIS 2.24% (**14** of 625) of the time). This confusion makes sense, as ablation analysis may often be discussed alongside analysis of results. Furthermore, the information unit RESULTS appears almost 3.65 times as often as the information unit ABLATION ANALYSIS, another example of how the uneven distribution of information units impacts the accuracy of our classifier.

Similarly, sentences describing EXPERIMENTS were incorrectly labeled as belonging to RESULTS 25.97% (**87** of 335) of the time, and RESULTS appears in the training data 3.16 times as often as EXPERIMENTS. This resulted in experiments sentences never being positively identified by our system in Phase 1. Lastly, our system was not able to positively identify sentences belonging to the information unit APPROACH, either classifying them as non-contributing, or as belonging to the information unit MODEL.

## 6.2 Implications for Future Work

Because the frequently confused sentences often have many features in common, in future work we will investigate a document-level classification approach, to explore whether there is information in the paper as a whole that can point towards one information unit over another. For example, the information units APPROACH and MODEL never appear together in the same article in the training data, because they describing the model used is equivalent to describing the approach taken. One might be able to determine whether a paper is likely to discuss a model rather than a general approach by looking at features of the whole document.

One type of feature that might be leveraged for better results is section headers. For example, sentence 5 in Table 3 is found in a section with the header “Our Approach”, which indicates the gold label of APPROACH. Often, the contribution sentences describing results are under a section header with the word “Results” in it, as is the case for sentence 3 in Table 3. Similarly, sentence 2 in Table 3 is found in a section labeled “Experimental Setup”. However, the correct information unit is not always consistent with the section header. Sentence 1 in Table 3 is labeled with the information unit HYPERPARAMETERS, though it is found in a subsection called “Set-Up” under the section “Experiment Results”. Sentence 4 in Table 3 is found in a section called Experimental Results, but has the gold label ABLATION ANALYSIS.

	Predicted	True	Sentence
1	Experimental Setup	Hyperparameters	The initial learning rate is 0.0004 and the batch size is 32 .
2	Hyperparameters	Experimental Setup	Word2vec is used to produce the word embeddings .
3	Ablation Analysis	Results	Furthermore , using pseudo entity annotations boosted the accuracy by 0.3 % .
4	Results	Ablation Analysis	The best performance is achieved by Faceness , with a recall below 20 % .
5	Model	Approach	Crucially , the BiLSTM is trained with the rest of the parser in order to learn a good feature representation for the parsing problem .

Table 3: Examples of mislabeled sentences

Error	Common Words (frequency)	Top POS (probability)
B-miss	our (124), all (17), to (14), each (14)	PRP\$ (.2255)
E-miss	model (20), to (17), using (13), method (8)	NN (.2411)
B-extra	best (3), use (2), set (2), the (2)	NN (.4235)
E-extra	. (24), on (8) in (7), than (7), by (6)	IN (.2784)

Table 4: For each error type, this table shows the most frequent words either missed by predicted phrase spans, or added (extra) by predicted phrase spans. Column 3 contains the most frequent POS tag for each error type (e.g. 22.55% of all words missed from the beginning of a span were possessive pronouns).

## 7 Entity and Relation Extraction Results

The gold test data had 13,028 total phrase spans, of which our system identified 4,277. 39.72% of these predicted phrase spans were exact matches with gold phrase spans. 29.97% of predicted phrase spans were complete false positives, not overlapping with any gold phrase spans. 30.30% of predicted phrase spans were not perfect matches, but did overlap with gold phrase spans. The majority of partial matches were missing a word, reflecting the fact that the average span length for the gold data was 12.37 characters but the average span length for the predicted data was 10.49 characters. 42.44% of partial matches were missing characters in the beginning of the phrase span, 52.11% were missing characters in the end of the phrase span, 6.56% had extra characters in the beginning of the phrase span, and 14.97% had extra characters in the end of the phrase span.

### 7.1 Partial Phrase Matches

In this section we first look at the characteristics of the partial matches to discover the cause of these errors. Because the Duluth system selected phrase spans using grammatical features, we look at the parts of speech of the words that were either missed by the Duluth system or erroneously added to phrase spans. Then, we look at the grammatical characteristics of the gold phrase spans that were entirely missed by the Duluth system.

The most common errors and their frequencies are shown in Table 4. The part-of-speech that was most likely to be missing from the end of a span (E-miss) in the Duluth system output was NN. Our system misses these noun phrases because the Duluth system outlines sentences starting with the verb labeled as ROOT by Stanford Core NLP’s dependency parser, and identifying the nsubj of that root and its dependencies as an entity <sup>2</sup>.

However, there are some cases where the word identified as the ROOT is not the verb that the subject of the sentence is directly dependent on. In these cases, the sentence subject goes undetected by our system. For sentence 1 in Table 5, if the verb *improves* were properly identified as the ROOT, the sentence might be outlined like so: Built on top of the model in but excluding ELMo, *base reinforced model* (ENTITY) *improves* (RELATION) the *average F1 score* (ENTITY). However, *Built* is identified by the dependency parser as the ROOT, so the Duluth system fails to extract the noun phrase *our base reinforced model*, which is the nsubj phrase dependent on *improves*. Notice also in this example the omission of the possessive pronoun *our* in the system outline; this illustrates the most common part-of-speech missing from the beginning of

<sup>2</sup>ROOT refers to the root of the dependency parse tree. The nsubj of a parse is the nominal subject dependent on the root.

1	Built on top of the model in but excluding ELMo, our base reinforced model improves the average F1 score around 2 points [...]	
1	System: base, built on, top	Gold: our base reinforced model, improves, the average F2 score, around, 2 points
2	Finally, the baseline model, EDA, is largely outperformed by all other examined methods.	
2	System: baseline model, outperformed by, other examined methods	Gold: EDA, largely outperformed, by, all other examined methods
3	We also compare with TagSpace ( Weston et al. , 2014 ) , which is a tag prediction model similar to ours [...]	
3	System: We, compare with, TagSpace	Gold: compare with, TagSpace (Weston et al., 2014), tag prediction model

Table 5: Example sentences with phrase spans improperly extracted by the Duluth system. The phrase spans are shown separated by commas.

a phrase span, as 24.11% of words missing from the beginning of phrase spans have POS tag PRP\$.

27.84% of words that were erroneously appended to the end of phrase spans by the Duluth system were prepositions (IN). This is because the Duluth system includes prepositions in verb phrases, while the gold data contains some phrase spans where the preposition is separated from the verb, and others where the preposition is included in the same phrase span as the verb. Sentence 2 in Table 5 shows the Duluth system incorrectly including the preposition *by* with the verb *outperformed*, while sentence 3 in 5 shows our system correctly including the preposition *with* with the verb *compare*.

Phrase (Frequency)	of (502), with (295), on (274), for (260), in (190), to (129), from (87), using (85), by (80), as (66), than (60), at (52), is (46), between (44), results (39), over (36), achieves (36), based on (32), outperforms (31), training (26)
--------------------	---

Table 6: Most frequently missed phrases.

Phrase spans that improperly capture additional words can have a ripple effect when the additional word is supposed to belong to a different phrase span, like the additional word *by* in the phrase span *outperformed by*. Because *by* is included in a phrase span already, it does not exist in its own phrase span as it does in the gold data, which means that the phrase span *by* is completely omitted by the Duluth system.

This error is quite common; 97% of the phrase spans in the gold data consisting of a single preposition were not identified by the Duluth system, due to the fact that these words tend to get absorbed by other phrases. This continues to have a detrimental effect further down the pipeline, as whether the prepositions are alone or chunked with other phrases affects the formation of triples.

## 7.2 Missed Phrases

To determine the possible causes of missing phrase spans, we looked at the error rates by part-of-speech. Since 77.31% of the total phrase spans in the gold standard data were missed by our system, we are only considering POS patterns that were missed over 77.31% of the time. We are also only looking at POS patterns that occurred at least 100 times in the gold data.

Phrase spans made up of a single preposition were frequently not identified by the Duluth system. This is apparent in Table 6; the top 5 phrases most commonly ignored by the Duluth system are all prepositions. Table 7 shows the phrase types (phrases described by the POS tag for each word) that are least likely to be captured in their entirety by the Duluth System. In addition to prepositions, infinitive verbs (TO VB) are also frequently omitted by the Duluth system. This is because the Duluth system bases the outline of the sentence off of the ROOT, which is almost always a finite verb. For this reason, infinitives are frequently ignored.

Our grammar-based approach might be improved on by changing the rule that outlines the sentences based on the ROOT verb of the sentence, in order to focus more on noun phrases that are more likely to be scientific entities. This would

POS Pattern	Missed/Total	Example
IN	.9735	for
TO	.9416	to
CD	.9364	99.60
TO VB	.9098	to compute
RB	.9040	jointly
VBG	.8837	using
NN NNS	.8750	feature maps

Table 7: These are the POS patterns that are most likely to be ignored by the Duluth system, and the conditional probability that they will be omitted.

require developing a method for tagging likely scientific entities that improves on our MEMM entity tagger. However, entities extraction might be better addressed with a neural approach since our grammatical approach (without semantics) does not capture the nuances in the training data.

## 8 Triples Extraction Results

The official F1 score on the gold standard test data for triples extraction from Phase 2 Part 2 was 0.2762, and the F1 score for information units was 0.7556. The Duluth system scored the lowest at extracting triples for the information units DATASET, TASKS, EXPERIMENTS, receiving F1 scores of 0.0, 0.0, and 0.0597 respectively. The extracted triples are input into files named for the information unit they belong to.

Triples extracted from a sentence classified as RESEARCH PROBLEM are output into a file named research-problem.txt. The gold data does not contain any tasks.txt files, so all tasks.txt files generated by our system were false positives. The gold data contained two dataset.txt files, both of which were missed by our system. These scores are consistent with the initial results from sentence classification, as seen in Table 1. Because our system deals with information unit classification during the sentences extraction step rather than the triples extraction step, we focus our discussion here on errors related to our triples extraction methodology.

Our triples extraction system relies heavily on part-of-speech tagging to determine whether an entity belongs to the edge of a triple or the middle (for example, noun-phrases are more likely to be subjects or objects, and verb-phrases are more likely to be predicates). In order to determine the efficacy of this approach, we look at the words that were improperly positioned in our system triples.

One pattern that emerged is that many words were wrongly positioned in the Duluth system data. Some phrases and their POS tags that only exist in the middle position in the gold data that are found in the edges of triples in the system data include “achieves” (VBZ), “propose” (VB), “performs” (VBZ), and “uses” (VBZ). Other phrases that only exist in the edges of triples in the gold data that are found in the middle position in the system data include “outperformed” (VBG), “worse” (JJR), “outperforming” (VBG), and “randomly initialized” (RB VBN).

This observation is consistent with the POS distributions found for the gold triples not identified by the system and the system triples that were wrongly identified; 70 of the 792 phrases belonging to the edge of the gold triples missed by the Duluth system have the POS pattern RB VBN, while 46 of the 730 phrases that were wrongly positioned in the middle of triples by the Duluth system also have the pattern RB VBN. Similarly, 324 of the 1,656 phrases belonging to the middle of gold triples missed by the Duluth system have the POS tag VBZ, while 712 of the 4,242 phrases that were wrongly positioned at the edge of triples by the Duluth system have the same POS tag of VBZ. This shows that the Duluth system sometimes shifts the phrases to the left or right of where they ought to be in the triple pattern.

## 9 Future Work

One weakness of this system is that the selection of contributing sentences only uses sentence-level information; the classifier misses useful contextual information such as headers and the predicted class of preceding and following sentences. Future work may be able to address this problem by fine-tuning BERT to classify sequences of sentences rather than isolated sentences (Cohan et al., 2019). Generally, a document-level approach could be beneficial in terms of capturing important context.

Another issue is that the end of the system does not have the ability to provide feedback to earlier parts of the pipeline; the only agency it has in terms of contributing sentence selection is the ability to discard a sentence provided by the sentence classifier. Future work could incorporate a neural entity classification model into the entity and triple extraction subsystem, which could be used to validate or invalidate the classification made at the sentence level (Rotsztein et al., 2018).

## References

- Waleed Ammar, Matthew Peters, Chandra Bhagavatula, and Russell Power. 2017. [The AI2 system at SemEval-2017 task 10 \(ScienceIE\): semi-supervised end-to-end entity and relation extraction](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 592–596, Vancouver, Canada. Association for Computational Linguistics.
- Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. [SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Danqi Chen and Christopher Manning. 2014. [A fast and accurate dependency parser using neural networks](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar. Association for Computational Linguistics.
- Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Dan Weld. 2019. [Pretrained language models for sequential sentence classification](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3693–3699, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. [SemEval-2021 task 11: NLPContributionGraph - structuring scholarly NLP contributions for a research knowledge graph](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. [SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers](#). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana. Association for Computational Linguistics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Wei Chen. 2021. [DeBERTa: Decoding-enhanced BERT with disentangled attention](#). In *2021 International Conference on Learning Representations*.
- Simon David Hernandez, Davide Buscaldi, and Thierry Charnois. 2017. [LIPN at SemEval-2017 task 10: Filtering candidate keyphrases from scientific publications with part-of-speech tag sequences to train a sequence labeling model](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 995–999, Vancouver, Canada. Association for Computational Linguistics.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Ed-dine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. [Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge](#). In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP ’19*, page 243–246, New York, NY, USA. Association for Computing Machinery.
- Xinyi Liu and Artit Wangperawong. 2019. [Transfer learning robustness in multi-class categorization by fine-tuning pre-trained contextualized language models](#). *arXiv preprint arXiv:1909.03564*.
- Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Prismatic Inc, Steven J. Bethard, and David Mcclosky. 2014. [The stanford corenlp natural language processing toolkit](#). In *In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. [Scikit-learn: Machine learning in Python](#). *Journal of Machine Learning Research*, 12:2825–2830.
- Jonathan Rotsztein, Nora Hollenstein, and Ce Zhang. 2018. [ETH-DS3Lab at SemEval-2018 task 7: Effectively combining recurrent and convolutional neural networks for relation classification and extraction](#). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 689–696, New Orleans, Louisiana. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on*

*Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.



## A Ethical Considerations

The use of knowledge graphs to process, organize, and display information comes with some ethical implications, due to the fact that such a system acts as an intermediary between human readers and academic research. Overall such a system, if integrated into academia, may benefit authors of scholarly work by making their articles easier to find. It may also benefit readers, by making the process of literature exploration more efficient. However, unintentional harm may arise if there is bias in the training used to annotate the articles. Since the Open Research Knowledge Graph is structured so that contributions are interconnected in the graph across papers, there may be a risk that different kinds of contributions are more easily found than others.

Another concern is whether such an infrastructure could affect how people design and report their research, and if that effect is harmful. If researchers know that their work will be read by a machine and then integrated into the knowledge graph according to its most likely contributions, they could be consciously or unconsciously motivated to favor some methods and terms over others, in an attempt to optimize the likelihood that their work is seen. There is also the potential for bias to become embedded in the machine reader, that could influence what kinds of researchers have work that is easily discoverable in the knowledge graph.

## B Task Data

Prior to Phase 1, most of the data used to train the models was extracted from the original training dataset, and most of the provided trial dataset was used for evaluation. Six of the eight articles in the trial data that contained the information unit TASKS were moved to the training dataset so that it would contain enough examples to learn the patterns associated with this information unit. These folders were folder 7 from machine-translation, folders 3, 4, and 8 from named-entity-recognition, folder 8 from question-answering, and folder 2 from text-classification. For all evaluation phases, the models were retrained on the combined training and trial datasets.

## C System Architecture

The system architecture is organized into three sections: preprocessing, training, and testing. The preprocessing section is responsible for :

1. extracting noun and noun phrases from the training data entities.txt files,
2. extracting each sentence from the training data and labeling them with their information unit (or 0 for non-contributing sentences), and
3. extracting sentences from the evaluation phase data and labeling each one with its file path and sentence index.

The training pipeline is responsible for fine-tuning the deBERTa model using the sentences extracted from the training data, and training the MEMM using the extracted sentences and the list of nouns extracted from the training entities files.

The testing pipeline is responsible for taking the sentences extracted from the evaluation phase data and labeling each sentence with their predicted information unit, entity tags, and dependency parse. The data is passed between each step of the pipeline in a dictionary, which is added to at each step.

## D Information Unit Class Distributions

Phase 1 Class Distributions	
None	.9011
Ablation Analysis	.0062
Approach	.0030
Baselines	.0083
Code	.0010
Dataset	.0010
Experimental Setup	.0091
Experiments	.0072
Hyperparameters	.0102
Model	.0166
Research Problem	.0123
Results	.0228
Tasks	.0012

Table 8: Distribution of classes in the 59,755 training sentences provided by the training data.

## E DeBERTa Hyperparameters

optimizer	AdamW
learning rate	5e-5
max sequence length	128
epochs	8
batch size	32

Table 9: Hyperparameters used to fine-tune deBERTa

# INNOVATORS at SemEval-2021 Task-11: A Dependency Parsing and BERT-based model for Extracting Contribution Knowledge from Scientific Papers

Hardik Arora<sup>†</sup>, Tirthankar Ghosal<sup>\*</sup>, Sandeep Kumar<sup>†</sup>, Suraj Patwal<sup>†</sup>, Phil Gooch<sup>‡</sup>

<sup>†</sup>Indian Institute of Technology Patna, India

<sup>\*</sup>Institute of Formal and Applied Linguistics, Charles University, Czech Republic

<sup>†</sup>(hardik\_1901ce15, sandeep\_1911mc12, suraj\_1911mt12)@iitp.ac.in

<sup>\*</sup>ghosal@ufal.mff.cuni.cz, <sup>‡</sup>phil@scholarcy.com

## Abstract

In this work, we describe our system submission to the SemEval 2021 Task 11: NLP Contribution Graph Challenge. We attempt all the three sub-tasks in the challenge and report our results. Subtask 1 aims to identify the contributing sentences in a given publication. Subtask 2 follows from Subtask 1 to extract the scientific term and predicate phrases from the identified contributing sentences. The final Subtask 3 entails extracting *triples* (subject, predicate, object) from the phrases and categorizing them under one or more defined information units. With the NLPContributionGraph Shared Task, the organizers formalized the building of a scholarly contributions-focused graph over NLP scholarly articles as an automated task. Our approaches include a BERT-based classification model for identifying the contributing sentences in a research publication, a rule-based dependency parsing for phrase extraction, followed by a CNN-based model for information units classification and a set of rules for triples extraction. The quantitative results show that we obtain the 5<sup>th</sup>, 5<sup>th</sup>, and 7<sup>th</sup> rank respectively in three evaluation phases. We make our codes available at <https://github.com/HardikArora17/SemEval-2021-INNOVATORS>

## 1 Introduction

Thousands of papers are published by the scientific community every day. It is now increasingly becoming difficult to browse the huge pool of papers to identify relevant work and thereby keep up with the latest research findings. Scientific literature is growing at an exponential rate and researchers today face the problem to identify the latest state-of-the-art contributions. Keeping track of recent advancements is becoming a tedious exercise, if not practically impossible. The Open Research

Knowledge Graph (ORKG) (Jaradeh et al., 2019) is posited as a solution to keeping track of research progress minus the cognitive overload that reading dozens of full papers imposes. It aims to build a comprehensive knowledge graph that publishes scholarly publications' research contributions per paper, where the contributions are interconnected via the graph even across documents.

As described in D'Souza et al. (2021), with the ORKG comparisons feature, researchers are no longer faced with the daunting cognitive ingestion obstacle from manually scouring through dozens of papers of unstructured content in their field. This process traditionally would take several days or months; using the ORKG contributions comparison tabulated view, the task is reduced to just a few minutes. Assuming the individual paper contributions are structured in the ORKG, they can then deconstruct the graph, tap into the aspects they are interested in, and can enhance it for their purposes. Further, they can select multiple such paper graphs and click a button to generate their tabulated comparison. This presents an opportunity to enhance content ingestion enabled via their fine-grained machine interpretability by transforming scholarly articles into knowledge-based information flows by representing and expressing information through semantically rich, interlinked knowledge graphs (Auer et al., 2018).

In this paper, we present our approach for the three sub-tasks in the NLP Contribution Graph Challenge. Our contribution are as follows:

1. Fine tuning BERT for contributing sentences (a set of sentences about the contribution in the article).
2. A rule-based approach for extracting scientific and phrases (a set of scientific terms and relational cue phrases extracted from the contributing sentences; for each paper ) using

dependency parsing.

3. A CNN-based architecture for classifying sentences to 12 information units followed by rules to generate triples (semantic statements that pair scientific terms with a relation, modeled toward subject-predicate-object RDF statements for KG building).

The rest of this paper is organized as follows: Section 2 briefly summarizes some related works similar to this task followed by the problem statement of this task. Section 4 describes the details of the data provided by the organizers. Section 5 and 6 presents the details of our model for all three phases of the task, including the structure and its implementations, along with results and experimental details. The conclusions and the directions for the future research are provided in Section 8.

## 2 Related Work

Although this is a relatively new challenge, we found some related investigations in the literature. [Vogt et al. \(2020\)](#) proposed a novel semantic data model for modeling the contribution of scientific investigations of three domains, viz. Medicine, Computer Science, and Agriculture. The model includes a schema of relevant concepts highlighting six core information units, viz. Objective, Method, Activity, Agent, Material, and Result. They introduced the idea of building blocks called Knowledge Graph Cells for its knowledge graph application.

[Gupta and Manning \(2011\)](#) introduced a new categorization of key aspects of scientific articles, which is (1) FOCUS: main contribution, (2) TECHNIQUE: method or tool used, and (3) DOMAIN: application domain. They extracted the aspects by matching semantic patterns to dependency trees and learn the patterns using bootstrapping. They also present a case study on the computational linguistics community using the three aspects extracted from its articles, verifying our system’s results and showing novel results for the dynamics and the overall influence of computational linguistics subfields.

[Hayashi et al. \(2020\)](#) introduced a new task of disentangled paper summarization to generate summaries for the paper contributions and the work context to help identify the key findings shared in articles.

[Rusu et al. \(2007\)](#) presented an approach to extracting subject-predicate-object triplets from En-

glish sentences. They used four different well-known syntactic parsers for English to generate parse trees from the sentences, followed by extraction of triplets from the parse trees using parser-dependent techniques. A machine learning approach has been used by [Dali and Fortuna \(2008\)](#) to extract subject-predicate-object triplets from English sentences. Support Vector Machine (SVM) is used to train a model on human annotated triplets, and the features are computed from three parser.

## 3 Problem Definition

The problems are defined by the shared task organizers.

1. For Phase-1 (End-to-end Pipeline), given a scientific paper we have to output contributing sentence  $S_1, S_2, S_3 \dots S_{|n|}$  (where  $n$  is the number of contributing sentences present in the document), scientific term and predicate phrases from the contributing sentences and finally triples information for particular information units.
2. For Phase-2, Part 1 (Phrases and Triples), we are provided with gold annotated contributing sentences and we have to output scientific terms and predicate phrases from the contributing sentences.
3. Lastly, for Phase 2, Part 2 (Triples Extraction), along with the gold annotated contributing terms, the gold-labeled scientific term and predicate phrases are also provided. We have to output the triplets information for particular information units.

Table 2 shows the three sub-tasks with an example.

## 4 Dataset Description

[D’Souza et al. \(2021\)](#) released the data for this task. We are provided with a training set of 55084 sentences, taken from 236 annotated papers from across 24 various fields in the NLP domain (such as natural language inference, question answering, sarcasm detection, etc.). Out of these, 5084 comes under the category of contributing sentences, and the remaining are non-contributing sentences. Triples are organized into three (minimum) up to 12 information units (Research Problem, Approach, Model, Code, Dataset, Experimental Setup, Hyperparameters, Baselines, Results, Tasks, Experiments, and Ablation Analysis). The detailed description of

Information unit	Description	Example
Research Problem	It determines the research challenge addressed by a contribution using the predicate has ResearchProblem. By definition, it is the focus of the research investigation, in other words, the issue for which the solution must be obtained.	A Question - Focused Multi- Factor Attention Network for Question Answering
Approach or Model	Essentially, this is the contribution of the paper as the solution proposed for the research problem.	"More specifically , unlike existing models where the query attention is applied either token - wise or sentence - wise to allow weighted aggregation , the Gated - Attention ( GA ) module proposed in this work allows the query to directly interact with each dimension of the token embeddings at the semantic - level , and is applied layer - wise as information filters during the multi-hop representation learning process ." & First , it is embedding - agnostic , meaning that one of the main ( and perhaps most important ) hyperparameters in NLP pipelines is made obsolete .
Code	It is the link to the software on an opensource hosting platform such as Gitlab or Github or on the author's website.	We compute a vector gate as a linear projection of the token features followed l Code is available at <a href="https://github.com/kimiyoung/fg-gating">https://github.com/kimiyoung/fg-gating</a> l ar Xiv: 1611.01724v2 [ cs.CL ] 11 Sep 2017
Dataset	This is another aspect of the contribution solution in the form of a dataset.	To address this , this paper introduces the Stanford Natural Language Inference ( SNLI ) corpus , a collection of sentence pairs labeled for entailment , contradiction , and semantic independence .
Experimental Setup or Hyperparameters	Includes details about the platform including both hardware (e.g., GPU) and software (e.g., Tensorflow library) for implementing the machine learning solution; and of variables, that determine the network structure (e.g., number of hidden units) and how the network is trained (e.g., learning rate), for tuning the software to the task objective. It is called Experimental Setup when hardware details are provided, otherwise Hyperparameters.	We used pre-trained 300D Glove 840B vectors to initialize the word embeddings . & This takes two days using Tensorflow and a single NVIDIA K80 GPU . provide an official evaluation script that allows us to measure F 1 score and EM score by comparing the prediction and ground truth answers .
Baselines	They are the listed systems that a proposed Approach or Model is compared against.	( 5 ) BM25 : BM25 is a bag - of - words retrieval function that ranks a set of reviews based on the question terms appearing in each review .
Results	The main findings or outcomes reported in the article text for the ResearchProblem.	Overall , we observe a significant improvement with all three configurations , effectively showing the benefit of training a QA model in a semisupervised fashion with a large language model .
Tasks	The Approach or Model, particularly in multi-task settings, are tested on more than one task, in which case, we list all the experimental tasks. The experimental tasks are often synonymous with the experimental datasets since it is common in NLP for tasks to be defined over datasets. And where lists of Tasks are concerned, the Tasks can include the ExperimentalSetup as a sub information unit.	All the above subtasks have been modeled as binary classification problems : kernel - based classifiers are trained and the classification score is used to sort the instances and produce the final ranking .
Experiments	It is a container information unit that includes one or more of the previous discussed units as sub information units. Can be combination of lists of Tasks, ExperimentalSetup and Results, or a combination of Approach, ExperimentalSetup and Results.	The temperature parameter ? of Gumbel - Softmax is set to 1.0 , and we did not find that temperature annealing improves performance .
Ablation Analysis	It is a form of Results that describes the performance of components in an Approach or Model.	In , we removed dense connections over both co-attentive and recurrent features , and the performance degraded to 88.5.

Table 1: Information units and their corresponding definitions <sup>1</sup>

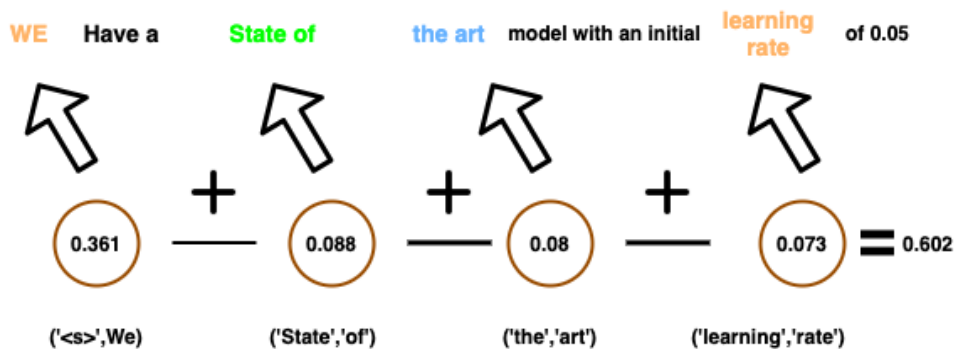


Figure 1: Bigram filtering to prune non-contributing sentences

Sentence:
We used the BERTBASE model pre-trained on English Wikipedia and BooksCorpus for 1M steps.
Scientific Term and Predicate Phrases:
used
BERTBASE model
pre-trained on
English Wikipedia
BooksCorpus
for
1M steps
Triples:
(C, has, ES)
(ES, used, BERTBASE model)
(BERTBASE model, pre-trained on, English Wikipedia)
(BERTBASE model, pre-trained on, BooksCorpus)
(BERTBASE model, for, 1M steps)

Table 2: Example of a Contributing Sentence, corresponding Scientific Term and Predicate Phrase, and extracted Triples

these information units is shown in Table 1. Overall, the annotated corpus contains 2631 triples (avg. of 52 triples per article). Its data elements comprise 1033 unique subjects, 843 unique predicates, and 2182 unique objects. Of all tasks, relation classification has the highest number of unique triples (544) and named entity recognition the least (473).

## 5 Proposed Approach

We describe our approach for all three phases of the competition as follows:-

### 5.1 Phase-I: Identifying Contributing Sentences

#### 5.1.1 Sentence Filtering

Initially, we use the Scholarcy API<sup>2</sup> to do some preliminary data analysis to understand some essential information (key concepts, highlighted sentences) in the challenge corpus.

To reduce the data-imbalance ratio of contributing and non-contributing sentences, we filter out most non-contributing sentences. We employ a simple bi-gram filtering to achieve this. We extract all the bi-gram pairs from the entire training corpus and assign each bi-gram pair a score (number of times the bi-gram pairs occurs in the corpus divided by 1000), based on which we set a threshold<sup>3</sup> and filter out the sentences. After filtering, 37.4% non contributing sentences are removed while only 7.07% of contributing sentences are filtered. The example in Figure 1 explains our approach.

*We have a state of the art model with an initial learning rate of 0.05,*

<sup>2</sup><https://www.scholarcy.com/>

<sup>3</sup>see our Github link mentioned in the abstract for details

Bigram tuples	Score
('<s>', 'We')	0.361
('<s>', 'The')	0.267
('of', 'the')	0.262
('on', 'the')	0.174
('in', 'the')	0.148
('<s>', 'In')	0.147
('to', 'the')	0.11
('with', 'the')	0.101
('and', 'the')	0.098
('state', 'of')	0.088
('the', 'art')	0.088
('with', 'a')	0.081
('the', 'model')	0.08
('our', 'model')	0.079
('learning', 'rate')	0.073
('<s>', 'Our')	0.071
('for', 'the')	0.068
('We', 'use')	0.068
('set', 'to')	0.064
('<s>', 'For')	0.063
('In', 'this')	0.058
('that', 'the')	0.058
('of', '%')	0.053
('word', 'embeddings')	0.053
('rate', 'of')	0.051
('natural', 'language')	0.05
('we', 'propose')	0.05
('is', 'a')	0.049
('from', 'the')	0.048
('this', 'paper')	0.047
('the', 'performance')	0.047
('based', 'on')	0.046
('<s>', 'To')	0.046
('is', 'set')	0.046
('question', 'answering')	0.044
('which', 'is')	0.044
('number', 'of')	0.044
('and', 'a')	0.042
('use', 'the')	0.042
('<s>', 'This')	0.04
('of', '< e >')	0.04
('batch', 'size')	0.04
('the', 'best')	0.039

Table 3: The topmost Bigram scores in non-increasing order; here(<s> denotes start token <e> denotes end token

From our generated list of bi-gram scores:

(< s >, 'We'), has a score of 0.308, ('state', 'of') has a score of 0.088 and so on. The total score is then calculated by summing the score of individual bi-grams.

#### 5.1.2 Classification Model

Input is the sequence of filtered sentences  $S = S_1, \dots, S_n$ , where  $n \leq 10$ . We set a threshold of 10 sentences on the number of sentences per sequence as released BERT pretrained weights support sequences of up to 512 word-pieces (Wu et al., 2016). The standard [CLS] is inserted as the first token of the sequence, and another delimiter token [SEP] is used for separating the segments.

After processing each sentence, we feed it into a SciBERT model (Beltagy et al., 2019) which is a variant of BERT, trained on scientific papers, as shown in Figure 2. The pre-training task of BERT (Devlin et al., 2019a) depends on two unsupervised sub-tasks: masked language modeling

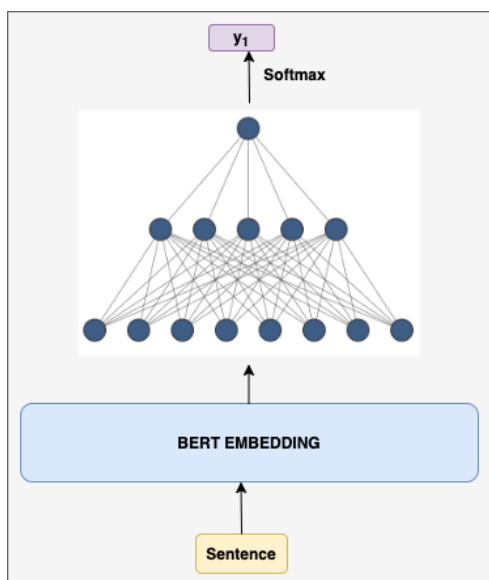


Figure 2: Classification of Contributing and Non-Contributing using a pre-trained SciBERT model

(MLM) and next sentence prediction(NSP). These two sub-tasks use the same model architecture but with different input patterns and different output layer. In MLM, a fixed amount of tokens of the input sequences is masked, and the model is trained for predicting the original tokens of the masked tokens. In NSP, the model has to predict whether two sequences of text are naturally following each other or not. 50% data is generated automatically by taking sentence pairs next to each other, and the other 50% is generated by taking sentence pairs randomly from the unlabeled corpus. The initial input embedding( $E_{Tok}$ ) is calculated by summing up the token, sentence and positional embedding. In the case of MLM, the final hidden vector of each of the masked tokens is passed to a softmax classifier(output layer) to predict the original token. On the other hand, during NSP, the final hidden vector( $C$ ) of the [CLS] token is fed to a binary classifier(output layer) to predict whether the input pair is following each other or not.

In the fine-tuning part for downstream tasks, we use the encoding of the [SEP] tokens to classify each sentence. The transformer layers (Devlin et al., 2019a) allows the model to fine-tune the weights of these special tokens according to the task-specific training data. We use a multi-layer feed forward network on top of the [SEP] representations of each sentence to classify them to the categories(is contributing or not?). During fine-tuning, the model learns appropriate weights for

Model	F1 (with filtering)	F1 (w/o filtering)
CNN+Glove	0.3123	0.1347
$Bert_{base}$	0.3578	0.1681
Our model	0.3987	0.1872

Table 4: Result of classification to contributing sentences, all are F1 scores

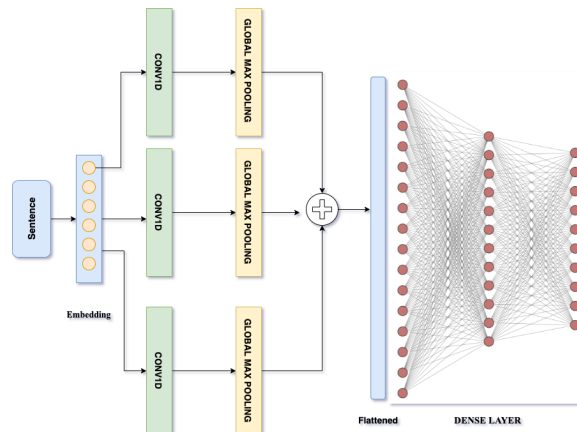


Figure 3: Architecture of classification of sentence into corresponding information units

the [SEP] token to capture contextual information and learn sentence structure and relations between continuous sentences (through the next sentence objective). Further, we use a softmax classifier on top of the MLP to predict the label’s probability.

### 5.1.3 Experimental Setup

We perform all our experiments on a GPU (GeForce RTX 2070) with 8 GB of memory. In phase-1, for contribution and non-contributing sentence classification task, we use the AllenNLP (Gardner et al., 2018) toolkit for the model implementation. As in prior work (Devlin et al., 2019b), for training we use the dropout of 0.1, the Adam optimizer for 2-5 epochs, and learning rates of  $5e-6$ ,  $1e-5$ ,  $2e-5$ , or  $5e-5$ .

## 5.2 Phase-II, Part 1: Phrase and Triples Extraction

Our system used an unsupervised rule-based system for extracting scientific entities and their predicates from the contributing sentences. As part of our initial experiments, we tried existing keyword extraction models such as RAKE, but they did not produce good results (e.g. F1 of 0.1062), as they are not tuned to this dataset. Given the paucity of training data for this task, we built a rule-based model for phrase extraction. We used the spacy

Phase	Avg F1	Sentences	Phrases span only	Information units	Triples + all units
Phase-1	0.3205	0.3987	0.1563	0.7172	0.0097
Phase-2, Part-1	0.5252	1.00	0.3740	0.7172	0.0097
Phase-2, Part-2	0.5971	1.00	1.00	0.3472	0.0413
<b>Top-performing System</b>					
Phase-1	0.4703	0.5941	0.4522	0.7293	0.1379
Phase-2, Part-1	0.7612	1.00	0.7857	0.8249	0.4344
Phase-2, Part-2	0.8594	1.00	1.00	0.8249	0.6129

Table 5: Our results for the three phases. Note: All scores in the table are F1 scores

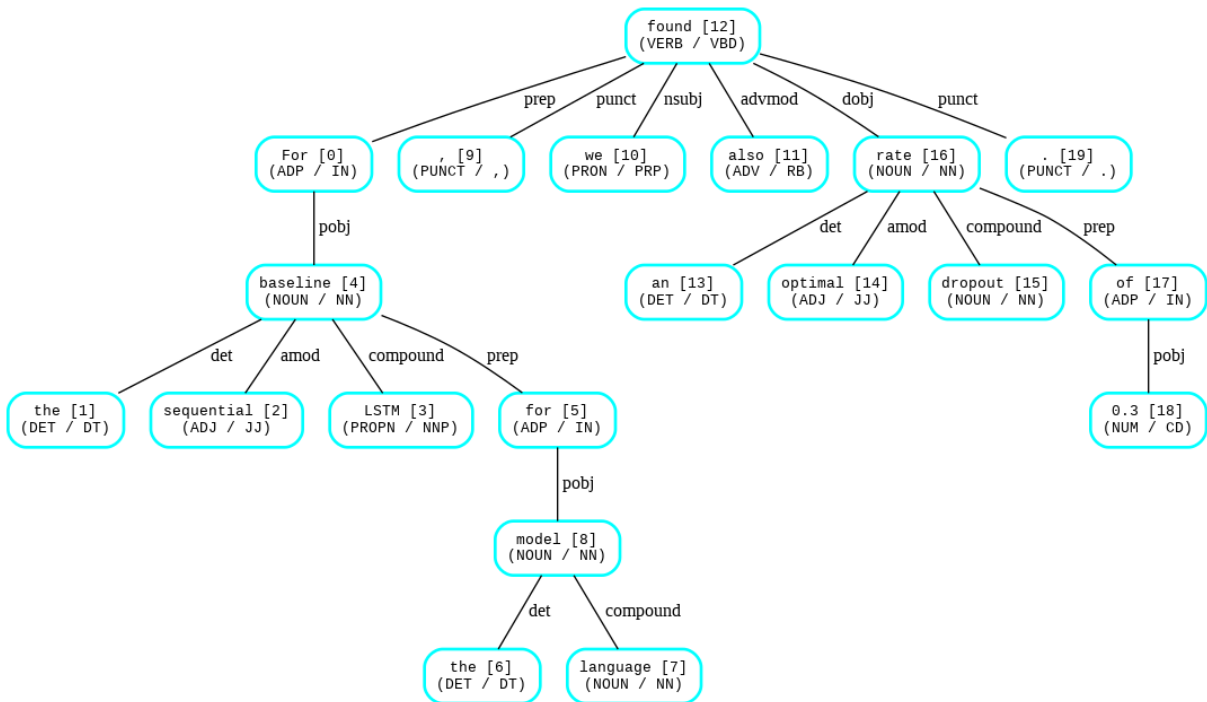


Figure 4: Example showing dependency parsing tree of a sentence

Non-contributing sentences misclassified as contributing sentences.	Reason
A model only achieved an F1 score of 86.5 on our development set, that is over 2 points lower than the 88.7 of a LSTM + A+D model.	Our model fails to differentiate between a general sentence that gives essential information but has no relation to the paper's model
As a by-product of our investigation, a variant of the RNNG without ensembling achieved the best reported supervised phrase - structure parsing ( 93.6 F1 ; English PTB ) and , through conversion , dependency parsing ( 95.8 UAS , 94.6 LAS ; PTB SD ).	Our model misclassified those sentences as contributing sentence which contains a large number of scientific terms
The elements of the stack that comprise the current constituent ( going back to the last 2 <a href="https://github.com/clab/rnng/tree/">https://github.com/clab/rnng/tree/</a>	It misclassified those sentences as contributing, which contains links that are not explicitly related to the paper

Table 6: Error analysis of Phase-1 (contributing sentence)

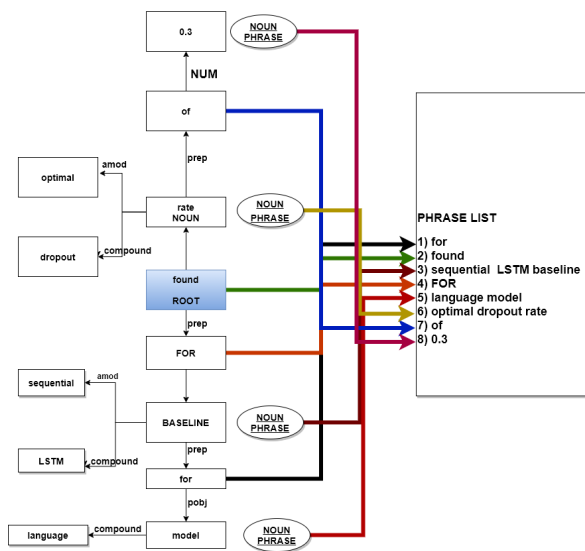


Figure 5: Illustration of addition of phrases to phrase list following rules for the sentence in Figure 4

True phrases	Predicted phrases
both models	models
tuned	tuned
dropout rate	dropout rate
to maximize	maximize
validation set likelihood	validation set likelihood
obtaining	
optimal rates	optimal rates
of	For
0.2 ( discriminative )	
0.3 ( generative )	

Table 7: Error analysis of Phase-2 (Phrase Extraction)

dependency parser<sup>4</sup> to generate a dependency tree for each contributing sentence. An example tree for the sentence, "For the sequential LSTM baseline for the language model, we also found an optimal dropout rate of 0.3," is shown in Figure 4. The rules specified are in such a format that considers some of the minute details, like if a word with "nsubj" dependency tag is a pronoun occurring at the start of the sentence, then the word should not be included in the phrase. We observed that the "ROOT" of the parsing tree (here: found) is primarily a constituent of the phrase list for that sentence, so we appended it. Next, we started exploring some particular child nodes of the "ROOT" node (mainly with dependency tags "dobj," prep," "advmod") as we tried to extract and form a proper noun, verb phrases, as shown in Figure 4. As soon as a child node with "NOUN" tag is found, complete noun phrase, is appended to the phrase list. At any level, if the child node (Cn) is a modifier (dependency tag 'prep'), then we will individually append it to the phrase list, followed by the subsequent exploration of the following hierarchy child nodes of Cn. This procedure continues recursively until we reach the leaf nodes for a branch. More details of various rules created by us for our unsupervised algorithm can be found in our shared source code.

### 5.3 Phase-II, Part 2: Triples Extraction

As there was little explicit section information in the provided parsed corpus, we classified each sentence into the 12 information units described above. We used a simple CNN-based neural architecture, and the model structure is shown in Fig 3. The first layer is the data input layer, followed by the embedded layer, which creates a numerical representation of the textual data. Then there are three parallel CNN models, each of which has a double convolution means a combination of a convolution layer and a pooling layer. To fully consider each word's information before and after, extract the size of the local characteristics of different sizes of 2x100,3x100,4x100. After the processing of the convolution layer, the characteristics of text classification are more advantageous. Based on this, the pooling layer is further screened from the global perspective where max pooling is used, followed by the merged layer, dense layer, dropout layer with a rate of 0.2. Finally, the text data is passed to a softmax function for outputting the classification

<sup>4</sup><https://spacy.io/>



result.

For triples creation, we created some rules. For research problems, triples are mainly in the form: "Contribution" followed by "has research problem" followed by the statement's scientific entity. For example:

*Contribution||has research problem||Text Comprehension*, is formed from the sentence "Gated - Attention Readers for Text Comprehension" belonging to a research problem.

We generate the triples in this format with the already extracted scientific phrases for the research problem information unit. Code triples are mainly in the form: "Contribution" followed by "code" followed by the URL of the available source code. For example:

*Contribution||Code||https://github.com/mandarjoshi90/coref*

We extract the URL using regular expression and generate the triples in the above format. We identified subjects along with their directly linked objects and predicates describing the relation between them for other triples. For example, if there is a sentence,

*ST Gumbel - Softmax estimator relaxes the discrete sampling operation to be continuous in the backward pass, thus our model can be trained via the standard backpropagation*

we form the triples as:

*(Contribution||has||Model),  
(Model||use||Straight-Through (ST) Gumbel-Softmax estimator)*

## 6 Evaluation

We report the evaluation results of all the three phases on Table 5. Average F1 score was to rank the participants in the competition. Precision, recall, and other details of the competition are available on leaderboard<sup>5</sup> of the competition. We also report the ablation analysis and the effect of filtering sentences in the extraction of contributing sentences of phase-1 in Figure 4. For phases-2 part-1, as were already provided by the gold label contributing sentences, the average F1-score of the contributing sentence is 1. Similarly, for phases-2 part-2, as were already further offered the gold label phrases, the average F1-score of the contributing sentence and the phases is 1. We reported a 0.32 average F1 for phase-1(end to end pipeline),

<sup>5</sup><https://competitions.codalab.org/competitions/25680#results>

0.52 for phase-2(phrases and triples extraction) and 0.59 for phase-3(triples extraction).

## 7 Error Analysis

As shown in Table 6 these sentences are non-contributing but misclassified as contributing sentences.

In phase 2, our model is based on rules, so it fails to capture unique or very different tree structures. For example as shown in Table 7, in sentence:-  
"For both models, we tuned the dropout rate to maximize validation set likelihood, obtaining optimal rates of 0.2 (discriminative) and 0.3 (generative).".

The phrases such as "0.3 (generative)" and "0.2 (discriminative)" was not captured. Our model also captured the single word phrase "maximize" instead of the true phrase "to maximize".

## 8 Conclusion and Future Work

With the NLPContributionGraph Shared Task, we have attempted to formalize the building of a scholarly contributions-focused graph over NLP scholarly articles as an automated task. Results and analysis on the gold test dataset show that our approach performed reasonably well in identifying contributing sentences and phrase extraction. However, we didn't perform well in triples extraction. In the future, we plan to improve the system, especially the triples extraction phase.

## References

- Sören Auer, Viktor Kovtun, Manuel Prinz, Anna Kasprzik, Markus Stocker, and Maria-Esther Vidal. 2018. [Towards a knowledge graph for science](#). In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, WIMS 2018, Novi Sad, Serbia, June 25-27, 2018*, pages 1:1–1:6. ACM.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [Scibert: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3613–3618. Association for Computational Linguistics.
- Lorand Dali and Blaz Fortuna. 2008. [Triplet extraction from sentences using svm](#). abs/2011.03161.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. [BERT: pre-training of](#)

- deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. Bert: Pre-training of deep bidirectional transformers for language understanding.
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. SemEval-2021 task 11: Nlpcontributiongraph - structuring scholarly nlp contributions for a research knowledge graph. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform.
- Sonal Gupta and Christopher D. Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Fifth International Joint Conference on Natural Language Processing, IJCNLP 2011, Chiang Mai, Thailand, November 8-13, 2011*, pages 1–9. The Association for Computer Linguistics.
- Hiroaki Hayashi, Wojciech Kryscinski, Bryan McCann, Nazneen Fatema Rajani, and Caiming Xiong. 2020. What’s new? summarizing contributions in scientific literature. *CoRR*, abs/2011.03161.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Ed-dine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019, Marina Del Rey, CA, USA, November 19-21, 2019*, pages 243–246. ACM.
- D. Rusu, Lorand Dali, B. Fortuna, M. Grobelnik, and D. Mladení. 2007. Triplet extraction from sentences.
- Lars Vogt, Jennifer D’Souza, Markus Stocker, and Sören Auer. 2020. Toward representing research contributions in scholarly knowledge graphs using knowledge graph cells. In *JCDL ’20: Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020, Virtual Event, China, August 1-5, 2020*, pages 107–116. ACM.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus
- Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

# MCL@IITK at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation using Augmented Data, Signals, and Transformers

Rohan Gupta    Jay Mundra\*    Deepak Mahajan\*    Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{rohangpt, jaym, dipakam}@iitk.ac.in

ashutoshm@cse.iitk.ac.in

## Abstract

In this work, we present our approach for solving the SemEval 2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). The task is a sentence pair classification problem where the goal is to detect whether a given word common to both the sentences evokes the same meaning. We submit systems for both the settings - Multilingual (the pair's sentences belong to the same language) and Cross-Lingual (the pair's sentences belong to different languages). The training data is provided only in English. Consequently, we employ cross-lingual transfer techniques. Our approach employs fine-tuning pre-trained transformer-based language models, like ELECTRA and ALBERT, for the English task and XLM-R for all other tasks. To improve these systems' performance, we propose adding a signal to the word to be disambiguated and augmenting our data by sentence pair reversal. We further augment the dataset provided to us with WiC, XL-WiC and SemCor 3.0. Using ensembles, we achieve strong performance in the Multilingual task, placing first in the EN-EN and FR-FR sub-tasks. For the Cross-Lingual setting, we employed translate-test methods and a zero-shot method, using our multilingual models, with the latter performing slightly better.

## 1 Introduction

A key challenge in lexical semantics is to identify or to encode the different senses of an ambiguous word. The Word Sense Disambiguation task (WSD) (Navigli, 2009) is a framework used to evaluate systems in their ability to identify different senses of the word. The task involves selecting the correct sense (meaning) of a target word from a list of senses listed in a sense inventory like WordNet (Fellbaum, 2012). Pilehvar and Camacho-Collados

(2018) proposed a novel benchmark (WiC - Word in Context Disambiguation) for the task casting the problem as a binary classification task, wherein it has to be identified whether a word common to a sentence pair is used in the same sense or not. The WiC task frees up the word sense disambiguation task from being tied to any sense inventory.

The SemEval 2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (Martelli et al., 2021) extends the WiC framework proposed by Pilehvar and Camacho-Collados (2018) to more languages. The task is divided into two subtasks - the Multilingual task and the Cross-Lingual task. The sentence pair, with a word in common, which is to be disambiguated, is drawn from the same language in the MultiLingual task, whereas the pair is drawn from two different languages in the Cross-Lingual Task. The task is posed as binary classification task over a pair of sentence wide contexts  $sent1$  and  $sent2$ , containing word sequences  $w_1$  and  $w_2$  respectively. The word sequences,  $w_1$  and  $w_2$ , have a common word in lemmatized form  $lemma$ . When  $w_1$  and  $w_2$  invoke the same sense of the lemma in their respective contexts, it is to be labeled as 'T' (True) class, else it labeled 'F' (False). As mentioned before, for the Multilingual setting,  $sent1$  and  $sent2$  are from the same language; for the Cross-lingual setting,  $sent1$  is from English, and  $sent2$  is from a non-English language. The languages considered for the Task are Arabic (AR), English (EN), French (FR), Russian (RU), and Chinese (ZH). Therefore for the Multilingual evaluation, we have AR-AR, EN-EN, FR-FR, RU-RU, and ZH-ZH settings, and for the Cross-Lingual evaluation, we have EN-AR, EN-FR, EN-RU, and EN-ZH settings. An example of Cross-Lingual sentence pair is given in figure 1. This task provides an evaluation benchmark for word sense disambiguation systems in languages other than English, a direction that has been less

\* Authors equally contributed to this work.

explored.

For this task, the training data is only provided for the EN-EN setting, and the development sets are provided only for the Multilingual task. As we are proposing supervised systems, it is essential to consider if we have training data or not. Therefore, splitting the Multilingual task, we propose systems for three components - (i) EN-EN (train and dev data available), (ii) Non-English Multilingual (only dev data available), and (iii) Cross-Lingual (neither train nor dev data available). Our models and implementations are available here<sup>1</sup>.

Click the right *mouse* button  
 le chat court après la *souris*  
 (French for The Cat follows the mouse)

Figure 1: An demonstrative example for the English-French Cross-lingual dataset. This pair will be classified as a 'False' pair.

## 2 Related Work

**Word Sense Disambiguation:** The techniques for the WSD task are broadly divided into knowledge-based and supervised approaches. The supervised approaches include fine-tuning BERT for sequence classification (Wang et al., 2019), EWISE (Kumar et al., 2019), and BiLSTM with attention (Raganato et al., 2017b). The knowledge-based methods use the information present in sense inventories such as WordNet (Fellbaum, 2012), BabelNet (Navigli and Ponzetto, 2012) and Wikipedia to derive semantic knowledge, assisting in the task of WSD. These include building sense embeddings for each sense of the word and disambiguate the target word using the nearest neighbour sense embedding: SensEmbBERT (Scarlini et al., 2020), (Loureiro and Jorge, 2019), or augmenting the pre-training objecting of BERT to take into account the sense information available in the WordNet: SenseBERT (Levine et al., 2019).

There are two existing benchmarks to evaluate the performance of WSD systems. One method is linked to the sense inventories, and the task is framed as a multi-class classification among the senses of a word listed in the inventory (Raganato et al., 2017a). The other is the WiC framework, not tied to any sense inventory, and asks if a target word has the same sense or not in the two given sentences.

<sup>1</sup><https://github.com/dipakamiitk/Crosslingual-WSD.git>

	Multilingual (EN-EN)	Multilingual (Others)	Cross-Lingual
Train Data	8000	✗	✗
Dev Data	1000	1000 (each)	✗

Table 1: Available data for this task. All datasets are balanced, that is, equal number of 'True' and 'False' pairs.

Recently, transformer-based architectures (Vaswani et al., 2017) (e.g., T5 (Raffel et al., 2019)) have outperformed all existing approaches when fine-tuned on the WiC task.

**Cross-Lingual NLP:** There are many NLP tasks for which the data is present in only some high resource language (often English), but the task needs to be solved for other languages. Some existing methods involve - (i) using multi-lingual language models (like mBERT (Devlin et al., 2018), XLM-R (Conneau et al., 2019)) to train on the high resource language and transfer the learning to other languages, or (ii) the translation approaches where we can translate train or test data to a target language and train a language-specific model. These methods have shown good performance on benchmarks like XGLUE (Liang et al., 2020) and XTREME (Hu et al., 2020), which have been designed to test this cross-lingual transfer performance of systems, with train data being present majorly in English, while the testing is to be done in other languages.

## 3 Corpus Description and Data Augmentation

A brief summary of the available data is shown in Table 1. This data is manually curated and covers four parts of speech - Nouns, Verbs, Adjectives, and Adverbs. In addition, we augmented the data by utilizing WiC, XL-WiC, and SemCor.

**WiC:** We use the data provided by the WiC task (Pilehvar and Camacho-Collados, 2018), which proposes the same problem as this task but only in English. They collected their data semi-automatically from WordNet (Fellbaum, 2012), and covered only Nouns and Verbs.

**XL-WiC:** It is a dataset that is an extension of the WiC dataset to multiple languages (equivalent to our multilingual setting) (Raganato et al., 2020). Again, the data was collected automatically from WordNet and Wiktionaries of various

languages. We are only interested in a few languages, from the WordNet development sets, the ones with good human performance, because we often note that this data does not accurately represent human distinguishable senses, which our manually collected task data set does. Specifically, we use Chinese(ZH), Danish(DA), Croatian(HR), and Dutch(NL). Farsi(FA) also has good human performance but we could not include it due to a pre-processing error.

**AuSemCor:** We created our own augmented dataset AuSemCor from the SemCor (Miller et al., 1993) dataset, which is a sense annotated corpora in English, with senses tagged using WordNet as its sense inventory. To generate data points for the same sense (T class), we pair up sentences containing a common lemma, whose WordNet senses are identical. For the other class (F class), we pair up sentences with a common lemma, but this lemma has different WordNet sense across the sentence pair. In addition, for the F class, we make sure that the WordNet supersense is also different for creating coarser sense distinctions as suggested by Pilehvar and Camacho-Collados (2018). We obtain 4986 datapoints with 2520 unique words. It is approximately balanced (2495 ‘F’ and 2491 ‘T’ pairs). Like WiC, it covers only Nouns and Verbs.

## 4 Proposed Approach

We shall now describe our proposed approaches for the two subtasks, multilingual and cross-lingual. We deal with English separately because training data is available in English and not in any other language. Since the data across the two tasks differ only in the language pairs, some general approaches apply to both settings. We finally submit an ensemble of models in all the tasks. The ensembling was done by taking average of the probability scores of the models (Probability Sum Ensemble).

### 4.1 Task Agnostic Proposals

**Signals:** We use a data preprocessing step of applying a signal to indicate the word to be disambiguated. This can be done in two ways - (i) *Signal 1:* encoding the target word (the word to be disambiguated) in both the sentences of a pair within double quotes (e.g., *Click the right “ mouse ” button*) as suggested by Huang et al. (2019), or (ii) *Signal 2:* append the target word at the end of the second sentence, similar to what was done by Wang et al. (2019). Note, for the former method;

we need the character spans of the target word to apply double quotes at the correct position.

**Sentence Reversal Augmentation:** For the models proposed in this task, the sentence pair is fed to the model in a manner such that the results do depend on which order the sentences are fed (i.e. the network parameters are not symmetric with respect to the two sentences). In such a case we propose the following augmentation - for every data point ( $sent1, sent2, lemma, label$ ), add another data point ( $sent2, sent1, lemma, label$ ) to the set of data points. A similar notion can be extended to making more robust predictions - at inference, before thresholding on the probability scores returned by the model, take into account the reversed sentence order, and average both the results. If such an averaging policy is followed for a particular model on the dev set, we follow the same policy on the test set. The *rev* subscript shall be used with dataset names to indicate that the data has been doubled using sentence reversal augmentation or with a model name to indicate that the model performs the reverse sentence averaging at inference for more robust predictions.

**Transformers+Logistic Regression:** Here we use the transformer-based pre-trained language model as an encoder network, feeding it with the sentence pairs concatenated with a separator token ( $[CLS]E(x_{sent1}^i[SEP]x_{sent2}^i[SEP])$ ). We then extract the word level embeddings (last layer hidden state) for each instance of the word (from both sentence 1 and sentence 2). If the word gets sub-tokenized, we pick the embedding of the first sub-token. We finally feed their concatenation to a logistic regression head, with binary cross-entropy as the loss function. The architecture can be seen in Figure 2. We used ELECTRA (Clark et al., 2020), ALBERT (Lan et al., 2019), XLM-R (Conneau et al., 2019) for English only data, and XLM-R for all other language data. Unless otherwise mentioned, we use the ‘*large*’ variant for ELECTRA and XLM-R, and the ‘*xxlarge*’ variant for ALBERT.

**Siamese Architecture:** Here, we cast our problem as a similarity problem. Similarity being measured by the closeness of the senses of the target word in each of the sentence. We, therefore, use a Transformer-based Pretrained Language Model to obtain the contextualized representations of the target word across the two sentences (using same model weights for both sentences), and optimize

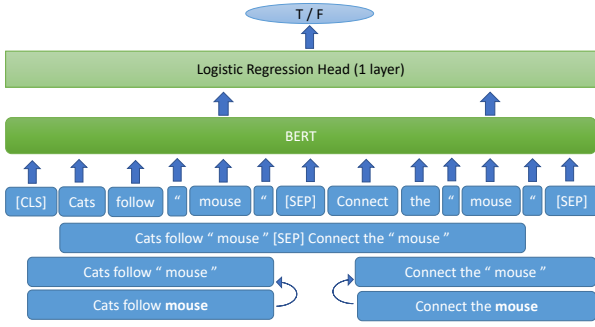


Figure 2: The Transformers + Logistic Regression Architecture (+ *Signal I*). This is our proposed architecture.

the contrastive loss (Hadsell et al., 2006) -

$$\mathcal{L} = \sum_i^{|T|} D_i^2 + \sum_j^{|F|} \max(0, m - D_j)^2$$

Where the set  $T$  is a set of same meaning sentence pairs, and  $F$  has different meaning pairs.  $D_i$  is the distance metric between the contextualized representations obtained for the target word by the language model for the pair of sentences. A small  $D_i$  would mean that the senses of the common word across the two sentences are the same.  $m$  is the margin parameter of the loss. We experiment with  $L_2$  and Cosine distances. We found this method to be good but not competitive with the Transformers+Logistic Regression method.

## 4.2 English (EN-EN) Task Proposal

For English language pair, we have dedicated training data (which we shall abbreviate as MCL-EN) that is used to train models. We used the proposed Transformers+Logistic Regression architecture and the preprocessing method of including signals using double-quotes.

In addition to sentence reversal augmentation, we augment the data using WiC and AuSemCor (Section 3). Both the WiC and the AuSemCor data have been automatically created using certain heuristics (Section 3), which make the sense distinctions in both the dataset a little different than what a human annotator would do. This is evidenced by the fact that human performance on the WiC benchmark is only 80%, while it is 97% on the Farsi dataset from the XL-WiC corpus, which was manually annotated. As this task’s data is human annotated, we do not use sentence reversal augmentation with WiC and AuSemCor data to avoid an

over-representation of the automatically annotated data in our training set.

## 4.3 Multilingual Task (except EN-EN) Proposal

For this task as well, we use Transformers+Logistic Regression architecture (the transformer being the multilingual XLM-R (Conneau et al., 2019)), with double-quote signal preprocessing and sentence reversal augmentation.

**Data:** For the four languages under this task, we have no training data. To address that, we split the development set for each of the language pairs into a 9:1 train-dev split. The split can be done in two ways - a random split or an out-of-vocabulary split (the 1 split will primarily have words not present in the 9 split). The latter’s motivation is to simulate the test set because the test set’s words will be unseen (not seen during training). The former may be useful as well because the model can see and learn more words during training. This distinction is based on Raganato et al. (2020)’s observation that models tend to perform better on seen words than unseen words during evaluation. We experiment using both split types. All languages’ data is concatenated together, and we solve the task for these four languages together by a single XLM-R model. We also use the EN-EN data of our task during this training. For development, we finally have  $100 \times 4$  data points, which we augment again by sentence reversal augmentation, thereby finally obtaining a dev set of 800 pairs. Since we perform sentence reversal on dev set, at test inference, we follow the reverse averaging policy as described before. This is denoted by a subscript *rev* in the model names. In our further discussion, we shall refer to this multilingual train data as MCL-MN<sup>rand</sup>, for the random split method, or MCL-MN<sup>oov</sup> for the out-of-vocabulary split method.

We augment our data using XL-WiC (Section 3). However, we do not use the XL-WiC data on all models, and whenever we do, we do not perform sentence reversal, to prevent higher representation of lower quality data. Sentence reversal is also not performed with English to have as much proportion of non-English data as possible in the training phase.

## 4.4 Cross-Lingual Task Proposal

For this task, we have no training or development data. We propose two methods - (i) Translate-Test, (ii) Multilingual Zero-Shot.

**Translate-Test:** In this method, we use Microsoft Translator<sup>2</sup> to translate the second sentence (in either AR, FR, RU, ZH) of the test set to English, thereby reducing it to an EN-EN task. However, this method is bound to introduce inaccuracies from translation, and we lose positional information about the target word in the translated sentence. So we cannot use word-level embeddings, and therefore the Transformer+Logistic Regression Model (Section 4.1) cannot be used. We, therefore tweak the Transformers+Logistic Regression architecture a little - we use the [CLS] token embedding instead of the word-level embeddings, keeping the Logistic regression head intact. For development, we use a back-translated EN-EN dev set (EN to FR to EN) for the second sentence to simulate inaccuracies of translation that will be induced in the second sentence in the test set. For training as well, we back translate 50% of the second sentence (12.5% to each AR, FR, RU, and ZH and back to en). Due to the loss of positional information about the target word after translation, we experiment with using *Signal 2* - appending the target word at the end of the second sentence. We use ELECTRA (Clark et al., 2020) as the encoder for this sub-task.

**Multilingual Zero-Shot:** In this method, we directly use the models obtained from our Multilingual task (section 4.3) on the Cross-Lingual test data.

## 5 Experiments and Results

We ran various experiments to test out the efficacy of the different approaches. All experiments have been carried out with a learning rate set at  $10^{-5}$ , using AdamW (Loshchilov and Hutter, 2017) optimizer, with batch sizes varying in  $\{8, 16, 32\}$ . We noticed that using a batch size of 32 was ideal. However, limited compute availability prevented us from trying it out. We trained our models for 10 epochs. To choose our best model, we performed validation multiple times during an epoch - 5 times an epoch for the EN-EN sub-task and 4 times an epoch for all other tasks. In all tables, we report accuracy, which is the task’s official evaluation metric.

The results and experimental set up of various models on the English set are summarized in Table 2. We obtain a total of 9 models for the EN-EN sub-task, four of them being *rev* variants. We submitted various ensembles, and the details are

<sup>2</sup><https://azure.microsoft.com/en-us/services/cognitive-services/translator/>

present in the lower half of the table. For the three listed ensembles, we violated our averaging policy; we do not average probability scores of the model with reverse sentence pair, even if the model was saved with such a policy on the dev set as these were overall best performing models on the dev set. At 93.3% on the test set, our model was the **best** performing model in the EN-EN task. Also noteworthy is an ensemble of models trained only on the data provided by the task, scoring 92.6% on the test set.

The results and experimental set up of various models on the Non-English Multilingual set are summarized in Table 5 for the development sets, and Table 3 for test set performance (where we show scores of various ensembles). As described in section 4.3, the OOV models refer to the models that were created by training on the out-of-vocabulary split method, while the RAND models refer to the models obtained by training on data created by the random split method. All models are based on XLM-R+Logistic Regression, with double quotes signal (indicated by + *Signal 1*). For evaluation on the test set, we ensemble a combination of models determined by the best performance on the joint dev set and language-wise dev set.

The performance of various translate-test models is shown in Table 4. We formed an ensemble of these models and submitted it to the leaderboard, which can be seen in Table 6. Also, in Table 6, we note that the best results are obtained by zero-shot application of the models trained in the multilingual sub-task to the cross-lingual sub-task.

## 6 Ablation Study

We perform an analysis of the various proposed approaches (Table 7), specifically paying heed to the EN-EN task, starting with a baseline of a BERT base model with a logistic regression head over the [CLS] token (the ‘cls’ models in the table). We see an improvement in performance by switching to target word embeddings (the last layer hidden state corresponding to the first sub-word token). Adding the signal in the form of double quotes (*Signal 1*) improves performance, probably by emphasizing the word to disambiguate, that, otherwise, the model does not know. *Signal 2* however, is not as effective, but it still improves performance over the non-signal model. Utilizing sentence reversal data augmentation, we see an improvement in performance. The model’s weights are not symmetric

Model	Trained On/Ensembled On	Dev	Test
XLM-R	MCL-EN <sub>rev</sub> + WiC	89.1	89.5
XLM-R <sub>rev</sub>	MCL-EN <sub>rev</sub> + WiC	89.3	89.5
XLM-R	MCL-MN <sub>rev</sub> <sup>oov</sup> + MCL-EN	89.1	89.9
ELECTRA	MCL-EN <sub>rev</sub>	<b>91.1</b>	90.3
ELECTRA <sub>rev</sub>	MCL-EN <sub>rev</sub>	90.8	91.7
ELECTRA	MCL-EN <sub>rev</sub> + WiC + AuSemCor	89.7	91.6
ELECTRA <sub>rev</sub>	MCL-EN <sub>rev</sub> + WiC + AuSemCor	90.5	90.9
ALBERT	MCL-EN <sub>rev</sub>	87.8	89.6
ALBERT <sub>rev</sub>	MCL-EN <sub>rev</sub>	89.7	<b>92.2</b>
Probability sum ensemble	All above models	<b>92.8</b>	<b>93.3</b>
Majority vote ensemble	All above models	92.7	<b>93.3</b>
Probability sum ensemble	Only MCL models	91.9	92.6

Table 2: Accuracies of the models on English (EN-EN) DataSet. All are + *Signal 1* models.

Submission	Ensemble Details	Dev	Multilingual Test			
			AR	FR	RU	ZH
1	OOV	88	84.5	86.2	86.1	86.4
2	OOV <sub>rev</sub>	88	84.4	87.5	85.4	85.6
-	OOV <sub>rev</sub> <sup>2</sup>	88.88	85.5	<b>87.8</b>	85.4	85.5
-	RAND <sub>rev</sub>	89.38	<b>86.0</b>	86.6	86.2	86.2
-	RAND <sub>rev</sub> <sup>2</sup>	90.5	85.7	86.7	<b>86.9</b>	86.0
-	Prob Sum (RAND <sub>rev</sub> <sup>2</sup> and OOV <sub>rev</sub> <sup>2</sup> )	NA	85.5	87.6	86.7	<b>87.3</b>

Table 3: Final Ensembles Non-English Multilingual. A “-” indicates model not submitted to the leaderboard.

Model	Accuracy
ELECTRA+ <i>Signal 2</i>	<b>86.4</b>
ELECTRA Back-T+ <i>Signal 2</i>	86.1
ELECTRA Back-T <sub>rev</sub>	85.6

Table 4: Translate Test Models evaluated on Back Translated EN-EN dev set. Back-T Models refer to models where 50% training data was also back translated.

Model	Trained on	Dev	Language-Wise Dev			
			AR	FR	RU	ZH
RAND XLM-R + <i>Signal 1</i>	MCL-EN+MCL-MN <sub>rev</sub> <sup>rand</sup>	87.38	89	85	91	96
RAND XLM-R + <i>Signal 1</i>	MCL-EN+MCL-MN <sub>rev</sub> <sup>rand</sup> +XL-WiC(ZH, DA, HR, NL)	88.13	88	84.5	91	95.5
OOV XLM-R + <i>Signal 1</i>	MCL-EN+MCL-MN <sub>rev</sub> <sup>oov</sup>	87	89	91.5	91	83.5
OOV XLM-R + <i>Signal 1</i>	MCL-EN+MCL-MN <sub>rev</sub> <sup>oov</sup> +XL-WiC(ZH, DA, HR, NL)	87	87.5	91	92.5	82

Note: The development sets for RAND and OOV models are different and hence are incomparable in performance.

Table 5: Non-English MultiLingual Development Set Accuracies. The Dev columns indicates the performance on the joint dev set, while the Language-Wise column lists down score for that particular language’s dev.

with respect to sentence 1 and 2, and ideally, we should train the model to lose its sense of order



Submission	Ensemble Details	Dev	Cross-Lingual Test			
			AR	FR	RU	ZH
-	OOV <sub>rev</sub>	-	<b>86.9</b>	<b>87.5</b>	87.6	<b>87.6</b>
-	OOV <sub>rev</sub> <sup>2</sup>	-	85.6	86.8	87.1	87.5
-	RAND <sub>rev</sub>	-	83.9	85.4	86.0	86.1
-	RAND <sub>rev</sub> <sup>2</sup>	-	85.4	86.7	86.9	84.5
-	Prob Sum	-	85.9	86.6	<b>88.0</b>	86.2
1	TT Ensemble 1	87.1	83.7	85.3	86.0	86.1
-	TT Ensemble 2	87.3	83.9	84.8	85.1	86.5
-	Adjusted Threshold RAND <sub>rev</sub>	-	<b>87.1</b>	<b>88.5</b>	<b>89</b>	<b>90.6</b>

Table 6: Final Ensembles Non-English Cross-Lingual Test. Translate Test models have been abbreviated as TT. The Dev column is only relevant to indicate the performance of TT models on the Back Translated EN-EN dev. The adjusted threshold model is purely for the purpose of analysis (Section 7).

and probably make better internal representations in the process. We do not observe a significant change in model performance on using the augmented data. In fact, for BERT<sub>base</sub>, it decays a little. Running models trained exclusively on WiC and AuSemCor, we note that the AuSemCor model performs better, but both models lag behind the model trained on MCL-EN. We note strong improvements in performance by ensembling various models (Table 2). The use of different transformers give us a diversity in our ensemble, with different models canceling each other’s mistakes. We also observe that the ensemble of models trained using only MCL data lag behind the ensemble of models trained using the augmented data (Table 2). This means that the augmented data is of benefit to our models. This was not clear with just a single model, BERT<sub>base</sub>, as mentioned before, with performance remaining around the same mark. As mentioned before, the Siamese models (83.5%) trail the Logistic Regression models (86.8%). Note that for Siamese models, the sentence order is irrelevant, so we cannot perform sentence reversal augmentation, and so 83.5% is their best with all our methods.

The analysis done on dev data is shown in Table 7. We finally took the models which were giving dev accuracy greater than 89% for the ensembles. That is XLM-R, ALBERT and ELECTRA. Another interesting point is that an XLM-R model trained for Non-English Multilingual subtask (using MCL-MN<sub>rev</sub> + MCL-EN) could also slightly improve itself (at the very least, it did not degrade) than when it trained on MCL-EN<sub>rev</sub> data.

For the Multilingual setting, we note that there is not much difference between the performance of OOV and RAND models (Table 3). The OOV

Model	Accuracy
cls BERT <sub>base</sub>	83.9
BERT <sub>base</sub>	84.6
cls BERT <sub>base</sub> +Signal 1	85.3
cls BERT <sub>base</sub> +Signal 2	84.3
BERT <sub>base</sub> +Signal 1	86.1
BERT <sub>base</sub> +Signal 1 (WiC)	72.6
BERT <sub>base</sub> +Signal 1 (AuSemCor)	77.5
BERT <sub>base</sub> +Signal 1 (MCL-EN + WiC + AuSemCor)	85.7
BERT <sub>base</sub> +Signal 1(MCL-EN <sub>rev</sub> )	86.8
BERT <sub>Large</sub> +Signal 1 (MCL-EN <sub>rev</sub> )	87.7
RoBERTa +Signal 1(MCL-EN <sub>rev</sub> )	87.7
XLM-R +Signal 1(MCL-EN <sub>rev</sub> + WiC)	89.1
XLM-R +Signal 1 (MCL-EN + MCL-MN <sub>rev</sub> <sup>rand</sup> )	89.3
ALBERT +Signal 1 (MCL-EN <sub>rev</sub> )	89.8
ELECTRA +Signal 1 (MCL-EN <sub>rev</sub> )	<b>91.1</b>
Sia. BERT <sub>base</sub> + Signal 1 , L <sub>2</sub> dist.	81.8
Sia. BERT <sub>base</sub> + Signal 1 , Cosine dist.	83.5

Table 7: An analysis of performance of models on EN-EN Dev. The training data was MCL-EN unless otherwise specified.

method works better for FR, while RAND works better for AR, RU, and ZH. On average, RAND scores are slightly better, but not by a large margin.

We note that some pre-trained language models perform better for this task, especially ELECTRA and ALBERT, which improve upon the scores of RoBERTa (Liu et al., 2019), and BERT. We also note that their ‘large’ variants are always better performers than the ‘base’ variants.

Actual \ Predicted	True	False
True	478	22
False	45	455

Table 8: Confusion Matrix of English (EN-EN) Sub-Task

Actual \ Predicted	True	False
True	1773	227
False	344	1656

Table 9: Confusion Matrix of Multi-Lingual (Non-English) Sub-Task (Model  $OOV_{rev}$ ).

## 7 Error Analysis

The confusion matrices for our best model are shown in Tables 8, 9, 10 and 11. On the Multilingual task, we found that number of false positives is higher than the number of false negatives for all languages.

In contrast, for the Cross-Lingual task, using the multilingual models in the zero-shot setting gives significantly lower false positives than false negatives (Table 10). The Translate-Test method offered a fairly balanced prediction, albeit with lower overall accuracy. (Table 11).

We observed that the probability scores returned by the multilingual models, when tested on the cross-lingual dataset, fell. We also observe a much lower number of false positives on this test data than the multilingual test data. This suggests that we need to tweak the prediction threshold value, in particular, to bring it down to adjust for the lower scores on this data set. Since we do not have a dev set for the cross-lingual sub-task, we perform the analysis on the test set itself. In Table 6, we can see a model’s performance in a threshold tuning experiment, where the threshold was brought down to 0.17 from 0.5 for all language pairs (i.e. 0.17 was used commonly for all language pairs). A significant spike in performance is observed (an average rise of 3.45% across all the cross-lingual language pairs). This suggests that models trained on multilingual data can competitively distinguish senses in the cross-lingual setting as well, provided we take into account the fall in probability scores, induced by the transfer, by threshold moving. As a control, a similar threshold tuning experiment for the same model on the multilingual test data

Actual \ Predicted	True	False
True	1627	373
False	131	1869

Table 10: Confusion Matrix of Zero-Shot model in Cross-Lingual Sub-Task (Model  $RAND_{rev}$ ).

Actual \ Predicted	True	False
True	1700	300
False	289	1711

Table 11: Confusion Matrix of Translate-Test model in Cross-Lingual Sub-Task (Model TT Ensemble 1).

POS	Accuracy
Adverb	86.67
Adjective	91.6
Noun	93.37
Verb	94.63

Table 12: POS wise accuracy analysis.

yielded only an average of 0.075% improvement for the four Non-English language pairs.

We also analyzed the Parts-Of-Speech (POS) wise performance of our English Model in Table 12. We see a lag in performance for Adverbs and Adjectives that have less number of training data points.

## 8 Conclusion

In this work, we presented our approach to solving the SemEval Task 2: Cross-Lingual and Multilingual Word in Context Disambiguation. We proposed different models based on transformers for the English, Non-English Multilingual and Cross-Lingual tasks. The application of signals and sentence reversal augmentation helped us improve performance across all tasks. Utilising the existing SemCor dataset, we created AuSemCor for the EN-EN sub-task. Due to the unavailability of training data in the Non-English Multilingual and Cross-Lingual task, we proposed methods to obtain the training data from the dev sets or external resources. We finally submitted ensembles for all tasks.

## References

- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Christiane Fellbaum. 2012. Wordnet. *The encyclopedia of applied linguistics*.
- R. Hadsell, S. Chopra, and Y. LeCun. 2006. [Dimensionality reduction by learning an invariant mapping](#). In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.
- Junjie Hu, Sebastian Ruder, Aditya Siddhant, Graham Neubig, Orhan Firat, and Melvin Johnson. 2020. Xtreme: A massively multilingual multi-task benchmark for evaluating cross-lingual generalization. *arXiv preprint arXiv:2003.11080*.
- Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. Glossbert: Bert for word sense disambiguation with gloss knowledge. *arXiv preprint arXiv:1908.07245*.
- Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. Zero-shot word sense disambiguation using sense definition embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Yoav Levine, Barak Lenz, Or Dagan, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2019. Sensebert: Driving some sense into bert. *arXiv preprint arXiv:1908.05646*.
- Yaobo Liang, Nan Duan, Yeyun Gong, Ning Wu, Fengei Guo, Weizhen Qi, Ming Gong, Linjun Shou, Daxin Jiang, Guihong Cao, et al. 2020. Xglue: A new benchmark dataset for cross-lingual pre-training, understanding and generation. *arXiv preprint arXiv:2004.01401*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Daniel Loureiro and Alipio Jorge. 2019. Language modelling makes sense: Propagating representations through wordnet for full-coverage word sense disambiguation. *arXiv preprint arXiv:1906.10007*.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- George A Miller, Claudia Leacock, Randee Teng, and Ross T Bunker. 1993. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.
- Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.
- Roberto Navigli and Simone Paolo Ponzetto. 2012. Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017a. [Word sense disambiguation: A unified evaluation framework and empirical comparison](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, Valencia, Spain. Association for Computational Linguistics.
- Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017b. [Neural sequence learning models for word sense disambiguation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark. Association for Computational Linguistics.
- Alessandro Raganato, Tommaso Pasini, Jose Camacho-Collados, and Mohammad Taher Pilehvar. 2020.

XI-wic: A multilingual benchmark for evaluating semantic contextualization. *arXiv preprint arXiv:2010.06478*.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. SenseBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *AAAI*, pages 8758–8765.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGlue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.

# HITSZ-HLT at SemEval-2021 Task 5: Ensemble Sequence Labeling and Span Boundary Detection for Toxic Span Detection

Qinglin Zhu<sup>1,†</sup>, Zijie Lin<sup>1,†</sup>, Yice Zhang<sup>1,†</sup>, Jingyi Sun<sup>1</sup>, Xiang Li<sup>1</sup>  
Qihui Lin<sup>1</sup>, Yixue Dang<sup>2</sup>, Ruifeng Xu<sup>1,‡</sup>

<sup>1</sup>Joint Lab of HITSZ-CMS, Harbin Institute of Technology(Shenzhen), China

<sup>2</sup>China Merchants Securities Co., Ltd

{zhuqinglin, 20S051050, xiangli, 1171000607}@stu.hit.edu.cn  
{lzjjeffery, zhangyc\_hit}@163.com, dangyixue@cmschina.com.cn  
xuruifeng@hit.edu.cn

## Abstract

This paper presents the winning system that participated in SemEval-2021 Task 5: Toxic Spans Detection. This task aims to locate those spans that attribute to the text’s toxicity within a text, which is crucial for semi-automated moderation in online discussions. We formalize this task as the Sequence Labeling (SL) problem and the Span Boundary Detection (SBD) problem separately and employ three state-of-the-art models. Next, we integrate predictions of these models to produce a more credible and complement result. Our system achieves a char-level score of 70.83%, ranking 1/91. In addition, we also explore the lexicon-based method, which is strongly interpretable and flexible in practice.

## 1 Introduction

41% of American adults in 2020 are reported experiencing some form of harassment<sup>1</sup>. Increasing incidents of online harassment and cyber violence have spurred researchers to investigate the problem of identifying and filtering offensive speech on the Internet. Most previously published insult detection tasks (Davidson et al., 2017; Xu et al., 2012) and methods (Aroyehun and Gelbukh, 2018; Modha et al., 2018) classify an entire comment (or document) to discern whether the comment is offensive or not, but cannot identify specific pieces of the toxic comment. Unlike previous studies, SemEval-2021 Task5: Toxic Span Detection(Pavlopoulos et al., 2021) requires the identification of the specific toxic spans, which is more innovative and challenging, and a key step towards a successful semi-automatic review of comments.

<sup>†</sup> Authors equally contributed to this work.

<sup>‡</sup> Corresponding Author: xuruifeng@hit.edu.cn

<sup>1</sup><https://www.pewresearch.org/internet/2021/01/13/the-state-of-online-harassment/>

More formally, toxic span detection is an extraction task, which is usually formalized as a Sequential Labeling (SL) problem, as shown in Figure 1(a), locating those spans by BIO tags. However, SL methods suffer from a huge search space due to the compositionality of labels (the power set of all sentence words), which has been proven in (Lee et al., 2016; Hu et al., 2019a). Therefore, in addition to SL formalization, we also formalize the task as a Span Boundary Detection (SBD) problem, as shown in Figure 1(b), locating those spans by start and end positions. Notice that, when there are multiple spans in a sentence, the matching of start and end positions may be ambiguous during decoding. This shows that theoretically, the SBD formalization is not consistently superior to the SL formalization. Hence, we choose to combine predictions of these two kinds of formalization to produce a more credible and complement result. Our system achieves a char-level score of 70.83%, ranking 1/91.

Besides, we also explore the lexicon-based methods, which usually have high precision but rather low recall, and are strongly interpretable and flexible in practice. First, we mine a toxic lexicon from the training set by a simple statistical strategy. Next, WordNet (Fellbaum, 2010) and GloVe (Pennington et al., 2014) are utilized to extend this lexicon further. With a toxic lexicon, we extract toxic spans through word-level matching.

## 2 Related Work

In recent years, cyber violence has become a widespread societal concern, and how to identify and filter hate speech has become an important topic in machine learning. TRAC proposes an aggression recognition task (Kumar et al., 2018) that provides a dataset of 15,000 annotated Facebook posts and comments in English and Hindi for



Figure 1: Comparison of SL and SBD, (a) denotes SL, (b) denotes SBD.

training and validation. The task aims to classify comments into three categories: non-aggressive, covertly aggressive, and overly aggressive. The Toxic Comment Classification Challenge 5<sup>2</sup> is an open competition in Kaggle that provides participants with comments from Wikipedia and defines six toxic categories: toxic, severe toxic, obscene, threat, insult, identity hate. In SemEval 2019 task 6 (Zampieri et al., 2019), in addition to whether the comment is offensive, the type of the attack and the target of the attack are also included. Based on this, Semeval 2020 task 12 (Zampieri et al., 2020) further extends the dataset to 5 languages: Arabic, Danish, English, Greek, and Turkish.

### 3 Methods

In the section, we describe how toxic span detection is formalized and corresponding solutions in detail.

#### 3.1 Sequence Labeling

The BIO tag scheme is utilized to locating toxic spans, where B (Begin) corresponds to the first token in a toxic span, I (Inside) corresponds to the inside and end tokens in a toxic span, and O corresponds to those no-toxic tokens. Following most existing work (Lample et al., 2016; Ma and Hovy, 2016), we leverage Conditional Random Fields (CRF) (Lafferty et al., 2001) for learning and inference.

In addition to token-level classification, CRF models the dependencies between tags in a tag sequence by the transition matrix  $A \in \mathbb{R}^{K \times K}$ , where  $K$  is the size of the tag space, i.e.  $K = 3$ . For the contextual representation  $\mathbf{x} \in \mathbb{R}^{n \times h}$ , the score of a tag sequence  $\mathbf{y} \in \mathbb{R}^n$  in CRF is defined as:

$$S(\mathbf{x}, \mathbf{y}) = h^1(y_1; \mathbf{x}) + \sum_{k=1}^{n-1} \left( h^{k+1}(y_{k+1}; \mathbf{x}) + A_{y_k, y_{k+1}} \right). \quad (1)$$

<sup>2</sup><https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

where  $h^k(y_k; \mathbf{x})$  is the score of the tag  $y_k$  at the  $k$  time step. Then, the conditional probability is obtained by a normalization operation:

$$P(\mathbf{y}|\mathbf{x}) = \frac{\exp(S(\mathbf{x}, \mathbf{y}))}{\sum_{\tilde{\mathbf{y}} \in \mathcal{Y}} \exp(S(\mathbf{x}, \tilde{\mathbf{y}}))}. \quad (2)$$

where  $\mathcal{Y}$  contains all possible paths of tag sequences. During inference, the predicted tag sequence  $\hat{\mathbf{y}}$  is obtained by:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \mathcal{Y}} P(\mathbf{y}|\mathbf{x}). \quad (3)$$

We adopt BERT(Devlin et al., 2019) and BERT+LSTM(Hochreiter et al., 1997) as the language encoder respectively, resulting in two solutions: BERT+CRF and BERT+LSTM+CRF. The reason for adding LSTM is that we believe that the contextual representation refined by LSTM could be more sensitive to the position of tokens.

#### 3.2 Span Boundary Detection

Different from SL formalization, SBD formalization utilizes the start and end positions tagging scheme to represent toxic spans. SBD formalization was originally applied in the machine reading comprehension task (Seo et al., 2016; Wang and Jiang, 2016). In these works, two  $n$ -classifiers are employed to predict the start position and end position separately, where  $n$  denotes the length of the input sentence. However, this strategy can only output a single span for an input sentence. Later, Hu et al. (2019b) extended the two  $n$ -classifiers strategy by a heuristic multi-span decoding algorithm. But this is not a concise and efficient solution for multi-span scenario, as the decoding algorithm relies on two hyper-parameters: (1)  $\gamma$ , the minimum score threshold, (2)  $K$ , the maximum number of spans. In addition to the two  $n$ -classifiers strategy, a more recent and popular strategy is to employ two binary classifiers to determine whether each token is the start (end) position or not (Li et al., 2020; Wei et al., 2020; Yu et al., 2019). In this paper, we adopt the binary classifiers strategy for SBD formalization and describe the details below.

Split	Train	Dev	Test
Num	6894	1723	2000

Table 1: Data statistics.

Given the contextual representation  $\mathbf{x} = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^{n \times h}$ , for the location  $i$ , we calculate the probability of whether it is a start position by Equation (4) and the probability of whether it is an end position by Equation (5).

$$p_{\text{start}}(i) = \sigma(W_1^\top x_i + b_1), \quad (4)$$

$$p_{\text{end}}(i) = \sigma(W_2^\top [x_i; p_{\text{start}}(i)] + b_2), \quad (5)$$

where  $W_1 \in \mathbb{R}^{h \times 1}$ ,  $W_2 \in \mathbb{R}^{(h+1) \times 1}$  and  $b_1, b_2 \in \mathbb{R}$  are model parameters.

The predictions of start and end positions are obtained by:

$$\text{starts} = \{i | p_{\text{start}}(i) > 0.5, i = 1, \dots, n\}, \quad (6)$$

$$\text{ends} = \{i | p_{\text{end}}(i) > 0.5, i = 1, \dots, n\}. \quad (7)$$

Then we adopt the nearest start-end matching strategy: for each predicted start position  $s \in \text{starts}$ , the nearest predicted end position  $e$  to the right of  $s$  is selected to form a predicted span  $(s, e)$ .

Similarly, we adopt BERT as the language encoder, and we call this model as BERT+Span.

### 3.3 Ensemble Strategy

Voting method is applied to integrate the results. In detail, for  $k$  different models, if no less than  $k/2$  models consider a character to be in the toxic span, the character is retained.

## 4 Experimental Setup

### 4.1 Data

The given trial data and training data are merged and the duplicates are removed. In addition, we fix some annotation errors, such as the partially-labeled words. 80% of the processed data is utilized for training and the rest is the validation set. Table 1 shows the statistics of the data used.

### 4.2 Parameter Settings

We find that the parameter size of the pre-trained model does not have a significant effect on performance, and therefore we simply adopt BERT-base as our language encoder, which consists of 12 transformer blocks with 12 representation heads. Three models are trained separately. The learning

	P(%)	R(%)	F1(%)
BERT+LSTM+CRF	71.99	89.96	69.34
BERT+CRF	74.50	88.10	69.44
BERT+Span	76.29	86.77	69.34
Ensemble	75.01	89.66	70.83

Table 2: Performance of three benchmark models and ensemble approach.

rate of BERT is set to  $2e-5$ , the learning rate of CRF is set to  $5e-3$ , and the maximum encoding length is 128. The weight decay is set to 0.01.

### 4.3 Evaluation Metrics

We use the official metric, i.e. char-level  $F1$ -score, as the evaluation metric. In addition, for a more detailed analysis, we also introduce character-level Precision ( $P$ ) and Recall ( $R$ ). Note that  $F1/P/R$  is the average over the samples, so there is no  $F1 = 2PR/(P + R)$ .

## 5 Results

### 5.1 Ensemble Approach

Table 2 shows the performance of three benchmark models and the ensemble approach. The experimental results show that all three models achieve similar results on  $F1$ -score, and integrating them results in an improvement of more than 1%, indicating that the predictions of the three models have good complementarity.

To further analyze the differences and respective advantages of SL and SBD formalization, we list their performances in single-span scenario and multi-span scenarios in Figure 2. It could be found that SBD formalization is more advantageous in single-span scenario, while SL formalization is more advantageous in multi-span scenario, which is consistent with our claim.

### 5.2 Lexicon-based Approach

We also explore a lexicon-based approach for predicting toxic spans. A toxic lexicon is mined from training data by a simple statistical strategy. More specifically, the toxic score of a word  $w$  is defined as below:

$$\text{toxic\_score}(w) = \frac{\#w\_in\_toxic\_span}{\#w\_in\_whole\_corpus}, \quad (8)$$

where  $\#w\_in\_toxic\_span$  is the count of appearances of word  $w$  in toxic spans, and  $\#w\_in\_whole\_corpus$  is the count of appearances

	# of words	P(%)	R(%)	F1(%)
Ensemble	-	75.01	89.66	70.83
Lexicon <sup>1</sup> (Wiegand et al., 2018)	551	75.13	44.47	33.07
Lexicon <sup>2</sup> (Wiegand et al., 2018)	2989	66.22	72.01	50.98
Lexicon <sup>original</sup> (Our)	119	76.71	82.22	64.98
Lexicon <sup>wordnet</sup> (Our)	231	72.56	84.05	64.09
Lexicon <sup>glove</sup> (Our)	186	73.98	83.34	64.19

Table 3: Results of Lexicon-based approaches and ensemble model on Precision, Recall and F1. Lexicon<sup>1</sup> and Lexicon<sup>2</sup> are two external lexicons. Lexicon<sup>original</sup> is collected by ourselves from training set. Lexicon<sup>wordnet</sup> and Lexicon<sup>glove</sup> are expanded from Lexicon<sup>original</sup> with WordNet and GloVe.

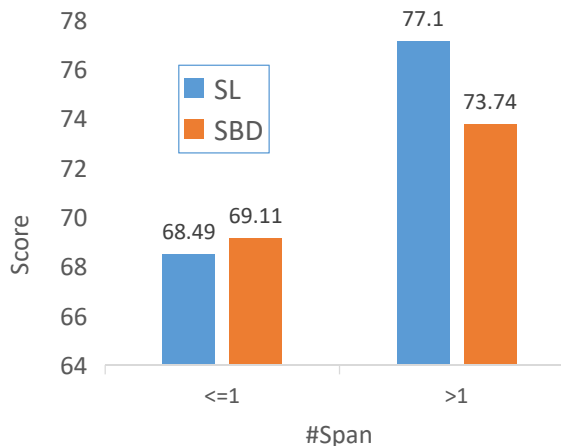


Figure 2: Comparison of the performance of SL and SBD method for data with different numbers of Spans.

of word  $w$  in the whole corpus. Then those words with a toxic score greater than a given threshold  $\theta$  are selected from a lexicon.

When predicting, the words in the sentence that appear in that toxic lexicon are extracted as the predicted toxic spans. There are three lexicons in our experiment, two of which were collected by (Wiegand et al., 2018), another is collected by ourselves from the training set.

Table 3 shows the results of the lexicon-based approaches and the ensemble approach, and we can observe that our lexicon-based approaches obtain notable results in the  $F1$ -score. In addition, we also calculate the average precision and average recall values of different methods on the test set, and our original lexicon-based approach even outperforms ensemble approaches in average precision, but there is still a significant gap in an average recall. Since the lexicon-based approaches can only identify the toxic words in the lexicon, the recall can be improved by expanding the toxic lexicon.

To improve the recall, we use WordNet (Miller, 1995) and GloVe (Pennington et al., 2014) to ex-

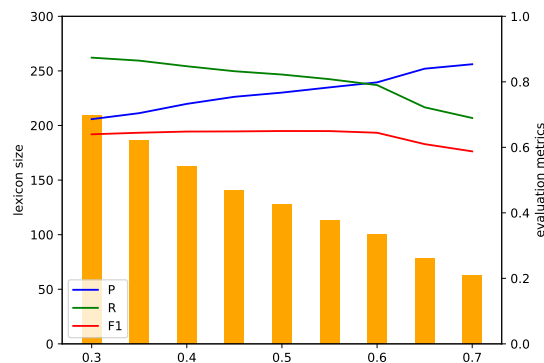


Figure 3: Performances of Lexicon<sup>original</sup> model with different threshold.

pand the toxic lexicon. In detail, we collect synsets of each toxic from WordNet, and collect the nearest similar words by calculating cosine similarity of GloVe vectors. The performances of the two expanded approaches are shown in Table 3. Although the recall of two approaches improves over the original lexicon, the precision decreases significantly, which indicating that there are a considerable number of non-toxic words in the synonyms found through WordNet.

Besides, we explore the impact of threshold  $\theta$  when mining the original lexicon on performance. The performances with different threshold is shown on Figure 4. As the threshold  $\theta$  increases, the size of lexicon decreases,  $P$  decreases,  $R$  increases,  $F1$  increases and then decreases, reaching a maximum 64.98 when  $\theta = 0.5$ .

## 6 Conclusion

In this paper, we formalize the toxic span detection as two problems separately and employ three state-of-the-art models. The strengths of each model are analyzed and a more credible and complement result is obtained through a voting approach. Our re-



sults achieve a good score (ranking 1/91). Besides, we explore a lexicon-based approach. The lexicon is mined from the annotation of the training data and then expanded by WordNet and Glove. Experiments show that the lexicon-based approach has not yet achieved the performance of the ensemble approach. We believe that future work could move towards combining deep learning-based methods and lexicon-based methods.

## Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (61632011, 61876053, 62006062), the Guangdong Province Covid-19 Pandemic Control Research Funding (2020KZDZX1224), the Shenzhen Foundational Research Funding (JCYJ20180507183527919 and JCYJ20180507183608379), and China Postdoctoral Science Foundation (2020M670912).

## References

- Segun Taofeek Aroyehun and Alexander Gelbukh. 2018. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 90–97.
- Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 11.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 4171–4186.
- Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243.
- Sepp Hochreiter, Jürgen Schmidhuber, and Corso Elvezia. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019a. Open-domain targeted sentiment analysis via span-based extraction and classification. *arXiv preprint arXiv:1906.03820*.
- Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li, and Yiwei Lv. 2019b. Open-domain targeted sentiment analysis via span-based extraction and classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 537–546.
- Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC)*, pages 1–11.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270.
- Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.
- Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020. A unified mrc framework for named entity recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074.
- George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.
- Sandip Modha, Prasenjit Majumder, and Thomas Mandl. 2018. Filtering aggression from the multilingual social media feed. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying (TRAC-2018)*, pages 199–207.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.
- Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.
- Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2020. [A novel cascade binary tagging framework for relational triple extraction](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1476–1488, Online. Association for Computational Linguistics.
- Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. [Inducing a lexicon of abusive words – a feature-based approach](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1046–1056.
- Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. [Learning from bullying traces in social media](#). In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 656–666.
- Bowen Yu, Zhenyu Zhang, Xiaobo Shu, Yubin Wang, Tingwen Liu, Bin Wang, and Sujian Li. 2019. Joint extraction of entities and relations based on a novel decomposition strategy. *arXiv preprint arXiv:1909.04273*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*.

# SarcasmDet at SemEval-2021 Task 7: Detect Humor and Offensive based on Demographic Factors using RoBERTa Pre-trained Model

**Dalya Faraj**

Department of Computer Science  
Jordan University of Science  
and Technology  
Irbid, Jordan  
dfalnore18@cit.just.edu.jo

**Malak Abdullah**

Department of Computer Science  
Jordan University of Science  
and Technology  
Irbid, Jordan  
mabdullah@just.edu.jo

## Abstract

This paper presents one of the top winning solution systems for task 7 at SemEval2021, "HaHackathon: Detecting and Rating Humor and Offense". The shared task 7 consists of two parts, task-1 with three sub-tasks 1a,1b, and 1c, and task-2. The goal of task-1 is to predict if the text would be considered humorous or not, then if it is yes, predict how humorous it is and whether the humor rating would be perceived as controversial. The goal of task-2 is to predict how the text is considered offensive for users in general. The proposed solution, SarcasmDet, has been developed using RoBERTa pre-trained model with ensemble techniques. The paper describes the submitted system's architecture with the experiments and the hyperparameter fine-tuning that led to this robust system. Our model ranked third and fourth places out of 50 teams in tasks 1c and 1a with F1-Score of 0.6270 and 0.9675, respectively. At the same time, the model ranked one of the top 10 models in task 1b and task 2 with RMSE scores of 0.5446 and 0.4469, respectively.

## 1 Introduction

In our daily life, the obstacles and difficulties in dealing with sarcasm, bullying, or even abuse of all kinds and ways are increasing day by day (Sheehan et al., 1999; Cleary et al., 2009; Tucker and Maunder, 2015; van Verseveld et al., 2021). Technically, sarcasm and bullying are among the most complex and challenging topics that major companies and institutes seek to address. Artificial intelligence and text processing techniques are the most potent current methods for detecting these problems within texts and images. Sarcasm and abuse are associated with attacking a specific person or group of people either through an unintended joke or, in many cases, by directly affecting the target's psyche. Irony and offensiveness are characterized by

their vocabularies that are peppered with humor to conceal the opposite (Lee and Katz, 1998).

Task 7 at SemEval-2021, "HaHackathon: Detecting and Rating Humor and Offense", provides two main tasks: task-1 with three sub-tasks (1a,1b, 1c) and task-2. The goal of task-1 is to predict if the text would be considered humorous or not, and if it is yes, then expect how funny it is and whether the humor rating would be perceived as controversial. The goal of task 2 is to predict how the text is considered offensive for users in general. Our solution, SarcasmDet, has been ranked among the top four teams in two sub-tasks. The proposed approach uses the provided dataset, which contains 10K of row text data. We have experimented with several pre-trained language models using the simple transformers library. It is worth mentioning that using the hard-voting ensemble technique has increased our score remarkably.

The paper is constructed as follows: Section 2 provides the related works. Section 3 and 4 describe the shared task and the provided dataset, respectively. Section 5 describes our system solution. Section 6 shows our experiments. Section 7 provides the results, and finally, the conclusion is in Section 8.

## 2 Related Works

In recent years, social media's development and growth have motivated the NLP research community to detect Humor and Offensiveness. In 2018, SemEval provided different shared tasks to detect emotions and irony in tweets (Mohammad et al., 2018; Van Hee et al., 2018). The top teams' proposed models mostly used LSTM and word embeddings (Abdullah and Shaikh, 2018; Badaro et al., 2018; Wu et al., 2018). In 2019, SemEval also introduced a shared task to discover offensive language in social media. Researchers in (Liu et al., 2019a)

used the dataset of the Offensive Language Identification Dataset (OLID) provided by (Zampieri et al., 2019). They ranked first in the task with an F1 (Macro) score of 0.8286 by applying linear model, LSTM, and BERT pre-trained model. In 2020, one of the shared tasks presented in SemEval was about how to change a chunk of text to make the text funnier. The authors (Mahurkar and Patil, 2020; Shatnawi et al., 2020) applied a pre-trained BERT model with different preprocessing for the presented dataset. This paper presents our solution to task 7 in SemEval2021, to detect humor and offensive simultaneously and explains it in detail.

### 3 Tasks Description

All subtasks of the SemEval2021 task 7 have different requirements. In this section, we have detailed the description for each task.

#### 3.1 Task 1a Humor Detection

Task 1a is a binary classification problem. The text should be classified as humor or not based on the answers of 20 participants to whether the particular text was intended to be funny or not. It is considered funny based on the majority of the participants' responses. Table 1 shows an example of the training dataset for task 1a.

#	Example	is humor
348	A baby's laughter can be the most beautiful sound you will ever hear. Unless it's 3 am. And you're home alone. And you don't have a baby.	1
6	Trabajo,' the Spanish word for work, comes from the Latin term 'trepariare,' meaning torture.	0

Table 1: Example for task 1a from the train dataset

#### 3.2 Task 1b Average Humor Score

Task 1b is a regression task; humor rating depends on the classified task 1a arguments. If the text was classified as funny (humor), then a question was raised about the level of humorous in the text on a scale of 1-5. Then, they took the average rating as a label. If not humorous text, they used 0 as a label. Table 2 shows an example of the training dataset for task 1b.

#	Example	humor rating
348	A baby's laughter can be the most beautiful sound you will ever hear. Unless it's 3 am. And you're home alone. And you don't have a baby.	3.1
15	Balsamic vinegar helps slowing the appearance of ageing signs healthy food health.	0

Table 2: Example for task 1b from the train dataset

#### 3.3 Task 1c Humor Controversy

Task 1c is a binary classification problem task; humor controversy depends on the classified arguments from task 1a. If the text was classified as funny (humor), then the task should determine whether the classification of the humor is controversial (1) or not (0). Table 3 shows an example of the training dataset for task 1c.

#	Example	humor controversy
348	A baby's laughter can be the most beautiful sound you will ever hear. Unless it's 3 am. And you're home alone. And you don't have a baby.	1
8000	Each ounce of sunflower seeds gives you 37% of your daily need for vitamin E vitamin health.	0

Table 3: Example for task 1c from the train dataset

#### 3.4 Task 2 Average Offensiveness Score

Task 2 is a regression problem task. The question was asked to determine whether the text is offensive in general, and how much general offensive is between 1-5. Table 4 shows an example of the training dataset for task 2.

### 4 Dataset Description

The dataset provided by (Meaney et al., 2021) SemEval 2021 organizers for task 7 contains 10,000 rows of text data and four columns of labels. The

#	Example	offense rating
27	How do the Chinese select their baby names? They chuck a tin can down the stairs Ping Wong ching Pang	3.8
1498	Today, I overslept and completely missed my 2nd nap.	0

Table 4: Example for task 2 from the train dataset

dataset is divided into three phases: training, development, and evaluation phase datasets. The dataset was collected by surveying US English native speakers of various ages between 18-70 and different genders, political situations, and income levels. The training set contains 8,000 rows of texts with four labels, every text in the data set have been classified based on four questions that were asked to the participants, and each question is related to a specific task. Each of the development set and the test set contain 1000 texts.

#### 4.1 Data Preprocessing

There was no need to implement preprocessing methods for the dataset of task1a and task2. However, the dataset for task1b and task1c contain null values. Therefore, we attempted to convert all null values into zeros, which lowered the data’s quality. Therefore, we used another technique, which is dropping the records with null. The later technique increased the data quality and gave better performances.

### 5 Systems Description

In our solution, we have used the pre-trained language model, RoBERTa (Liu et al., 2019b), that uses a robustly optimized NLP method to improve the Bidirectional Encoder Representations from Transformers. We have also used the BERT pre-trained model (Devlin et al., 2018). RoBERTa is built based on BERT’s language masking strategy, which learns to predict knowingly hidden sections of text within unannotated language examples. We have chosen RoBERTa pre-trained model because of the significant improvements in the performance by tuning the BERT training procedure and the architecture based on BERT-large. We have experimented with several deep learning models. In our

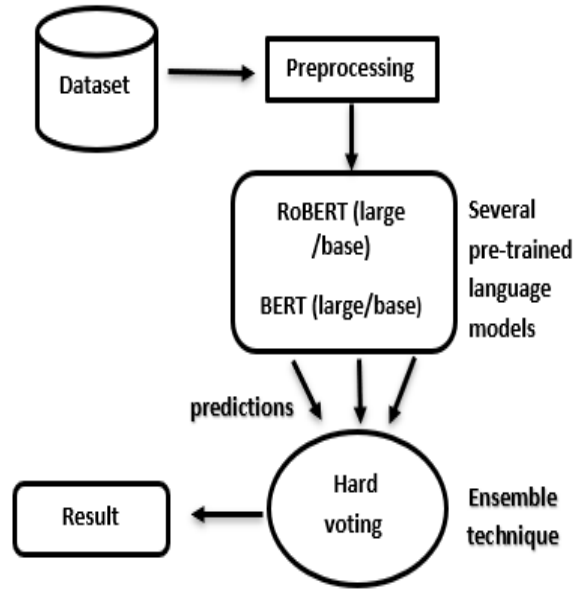


Figure 1: The architecture of our model.

best-performed solution system, we implemented ensemble technique (Chou et al., 2009) on the best-scored models that include RoBERTa-large, BERT-large trained on 24-layer, 1024 hidden, 16-heads, 355M parameters. We used RoBERTa model from HuggingFace(Wolf et al., 2019) and simpletransformers pre-trained models. More details about each subtask are as follows.

#### 5.1 Task 1a

We have applied RoBERTa (base/large) with different hyperparameters. Then, we have utilized the hard-voting ensemble technique to produce the best model that predicts the label in the test dataset. Our approach’s best result has scored 0.9513 F-score in the development phase and 0.9675 F-score in the test phase. The learning rate=1e-5, manual seed= 17, train batch size= 16, and num train epochs= 5.

#### 5.2 Task 1b

In this sub-task, we have applied BERT(base/large) cased with different hyperparameter. Then applied the hard-voting ensemble technique for the best model to predict the label in the test set (learning rate=1e-5, manual seed= 17, train batch size= 16, and num train epochs= 5).

#### 5.3 Task 1c

In this sub-task, we have applied several pre-trained NLP models, such as BERT(base-large), XINet(large), and RoBERTa(large), but the best solution was obtained from the two previous sub-

tasks (task 1a, task 1b). We used the best results of task 1a and task 1b to predict task 1c. If the result from task1a is 1, then that indicates it is humor. If the value of the Humer\_Ratting is equal or more than 3, then we consider that humor\_controversy to be 1. Otherwise, we assume humor\_controversy is 0.

#### 5.4 Task2

We have applied RoBERTa (large/base) with different hyperparameters. Then, we have used the hard-voting ensemble technique for the best model to predict the label in the test dataset (learning rate=1e-5, manual seed= 17, train batch size= 16, and num train epochs= 5).

## 6 Experiments

We have experimented with several pre-trained NLP models to detect Humor and Offensive through the development and evaluation phases. The pre-trained models include BERT(base/large) that is developed by Google researchers. Also, ALBERT(base/large)(Lan et al., 2019), which is a lite version of BERT to reduce parameters and increase the model speed by reducing memory consumption. Another pre-trained model is XLNet(base/large)(Yang et al., 2019), which introduced the automatic regressive pre-training method and outperformed BERT model in several tasks sentiment analysis, question answering and others. Finally, RoBERTa model, which outperformed most of the pre-trained models, if not all. We have implemented our experiments on google Colab using CPU and GPU. Using colab GPU increased the speed of the experiments by 100%. We used simple transformers library with various hyperparameters, learning rate=1e-5, manual\_seed= 17, train\_batch size=8-16-32 and epochs= 2-3-5. Our best results accomplished on all tasks was using hard-voting ensemble technique on top of best-scored results by RoBERTa-large and BERT-large-cased. The use of hard-voting technique increased the performance, and the accuracy, remarkably. In the development phase, our model ranked first place in three task1a, task1c, task2, and second place in task1c. However, for the evaluation phase, we have ranked 4th place in task1a, and 3rd place in task1c 3rd. Table 5 shows details of all hyperparameters used on all models for two phases.

Model	Epoch	Batch Size	LR	Manual Seed
RoBERTa-large	3,5	8,16,32	1e-5	17
RoBERTa-base	3,5	8,16	1e-5	17
BERT-base	2,3,5	8	1e-5	11,17
BERT-large	3,5	8,16,32	1e-5	11,17
XLNet-base	2,3	8,16	1e-5	17
XLNet-large	3,5	16	1e-5	17
ALBERT-base	2,3	8,16	1e-5	17
ALBERT-large	3,5	16	1e-5	17

Table 5: Hyperparameter was used in all experiments for two phases(development and evaluation) of all tasks.

## 7 Results

Our solution system results are divided into two phases development and evaluation phase. We experienced several pre-trained language models (RoBERTa, BERT, ALBERT, and XLNET) and implemented them using a simple transformers library in the development phase. In the evaluation phase(test phase), we improved our system solution's capabilities by using different hyperparameters with ensemble techniques. In the following sub-sections, we provided in detail all the results of the evaluation phase.

### 7.0.1 Task 1a

In task1a RoBERTa-large model outperformed all models with 0.9669 F-score and 0.9590 accuracies. We used the hard-voting ensemble technique to improve our results using the top best five achieved scores by RoBERTa-large and RoBERTa-base model with different hyperparameters. We have increased our solution performance and accomplished 4th place with a 0.9675 f-score and 0.9600 accuracies using this method. Table 6 shows the ensemble results and the top best results for RoBERTa model.

### 7.0.2 Task 1b

In task1b BERT-large-cased outperformed all models with 0.5468 RMSE. We improved the result

#	model	LR	Batch Size	F-Score	Accuracy
(1)	RoBERTa-base	1e-5	16	0.9654	0.9570
(2)	RoBERTa-base	1e-5	8	0.9660	0.9580
(3)	RoBERTa-large	1e-5	32	0.9661	0.9580
(4)	RoBERTa-large	1e-5	16	0.9663	0.9580
(5)	RoBERTa-large	1e-5	8	0.9669	0.9590
(6)	*	*	*	<b>0.9675</b>	<b>0.9600</b>

Table 6: Top best experiments used for task 1a in the evaluation phase by RoBERTa(large/base) model.\*Ensemble for 1,2,3,4,5

with the same method of ensemble and hyperparameters used in the previous sub-task. We have used the top best four achieved scores by BERT-large-cased and BERT-base-cased model, and we achieved 10th place with 0.5446 RMSE. Table 7 shows ensemble results and the top best results for the BERT model.

#	model	LR	Batch Size	RMSE
(1)	BERT-base	1e-5	8	0.5492
(2)	BERT-base	1e-5	16	0.5475
(3)	BERT-large	1e-5	16	0.5468
(4)	BERT-large	1e-5	8	0.5498
(5)	*	*	*	<b>0.5446</b>

Table 7: Top best experiments used for task 2 in the evaluation phase by BERT(large/base). \* Ensemble for 1,2,3,4

### 7.0.3 Task 1c

In task1c, we have implemented a method consisting of top of the best task1a and task1b results and using it. We accomplished third place with a 0.6270 F-score and 0.4699 Accuracy.

### 7.0.4 Task 2

Task2 RoBERTa-large outperformed all other models with 0.4559 RMSE. We have used hard-voting

ensemble technique and various hyperparameters with the top five best results by RoBERTa-large and RoBERTa-base. Using this technique, we achieved 10th place with 0.4469 RMSE. Table 8 shows ensemble results and top best results.

#	model	LR	Batch Size	RMSE
(1)	RoBERTa-base	1e-5	8	0.4828
(2)	RoBERTa-large	1e-5	32	0.4741
(3)	RoBERTa-large	1e-5	16	0.4609
(4)	RoBERTa-large	1e-5	8	0.4559
(5)	*	*	*	<b>0.4469</b>

Table 8: Top best experiments used for task 2 in the evaluation phase by RoBERTa(large/base) model. \* Ensemble for 1,2,3,4

## 7.1 Error Analysis

Our model was able to predict well in task 1a with an F1-score of 0.9675, but in task 1c, the prediction decreased with an F1-score of 0.52. Figures 2 and 3 show the confusion matrix for tasks 1a and 1c. The reason for this is due to the distribution of the datasets. In task 1a, the dataset was balanced, but in task 1c, the dataset was imbalanced as it contained null values.

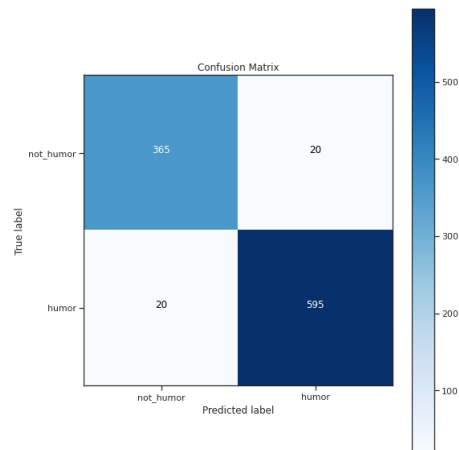


Figure 2: confusion matrix for task 1a.

## 8 Conclusion

This paper presents and describes our solution system for the SemEval2021 Task7: HaHackathon

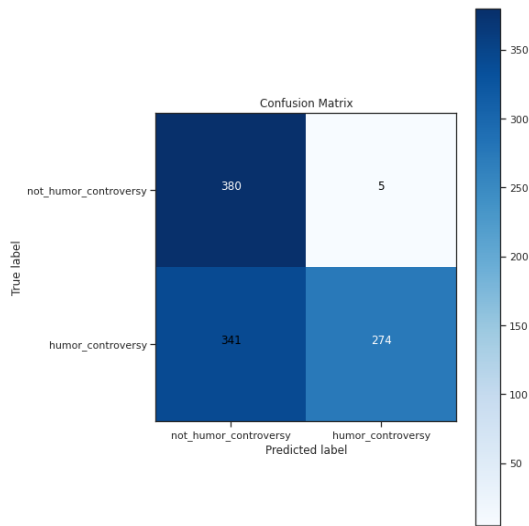


Figure 3: confusion matrix for task 1c.

detecting and rating humor and offense. We have applied several pre-trained language models, such as RoBERTa, BERT, ALBERT, and XLNET, with hard-voting ensemble technique to detect humor and offense mechanism. Our final solution was based on BERT-large-cased model and RoBERTa-large model, which showed remarkable improvements and a high overall outperformance. Our solution system ranked 4th place in task1a with a 0.9675 F-score, 10th place in task1b with a 0.5446 RMSE, 3rd place in task1c with a 0.6270 F-score, and 10th place in task2 with a 0.4469 RMSE.

## References

- Malak Abdullah and Samira Shaikh. 2018. Teamunc at semeval-2018 task 1: Emotion detection in english and arabic tweets using deep learning. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 350–357.
- Gilbert Badaro, Obeida El Jundi, Alaa Khaddaj, Alaa Maarouf, Raslan Kain, Hazem Hajj, and Wassim El-Hajj. 2018. Ema at semeval-2018 task 1: Emotion mining for arabic. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 236–244.
- Te-Shun Chou, Jeffrey Fan, Sharon Fan, and Kia Makki. 2009. Ensemble of machine learning algorithms for intrusion detection. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 3976–3980. IEEE.
- Michelle Cleary, Glenn E Hunt, Garry Walter, Michael Robertson, et al. 2009. Dealing with bullying in the workplace: Toward zero tolerance. *Journal of Psychosocial Nursing and Mental Health Services*, 47(12):34–41.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Christopher J Lee and Albert N Katz. 1998. The differential role of ridicule in sarcasm and irony. *Metaphor and symbol*, 13(1):1–15.
- Ping Liu, Wen Li, and Liang Zou. 2019a. Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 87–91.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Siddhant Mahurkar and Rajaswa Patil. 2020. Lrg at semeval-2020 task 7: Assessing the ability of bert and derivative models to perform short-edits based humor grading. *arXiv preprint arXiv:2006.00607*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17.
- Fara Shatnawi, Malak Abdullah, and Mahmoud Hamad. 2020. Mlengineer at semeval-2020 task 7: Bert-flair based humor detection model (bfhumor). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1041–1048.
- Michael Sheehan, Michelle Barker, and Charlotte Rayner. 1999. Applying strategies for dealing with workplace bullying. *International journal of manpower*.
- Emma Tucker and Rachel Maunder. 2015. Helping children to get along: teachers’ strategies for dealing with bullying in primary schools. *Educational Studies*, 41(4):466–470.
- Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50.



- Marloes DA van Verseveld, Minne Fekkes, Ruben G Fukkink, and Ron J Oostdam. 2021. Teachers' experiences with difficult bullying situations in the school: An explorative study. *The Journal of Early Adolescence*, 41(1):43–69.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. Thu.ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 51–56.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

# UPB at SemEval-2021 Task 8: Extracting Semantic Information on Measurements as Multi-Turn Question Answering

Andrei-Marius Avram<sup>1,2</sup>, George-Eduard Zaharia<sup>1</sup>,  
Dumitru-Clementin Cercel<sup>1</sup>, Mihai Dascalu<sup>1</sup>

University Politehnica of Bucharest, Faculty of Automatic Control and Computers<sup>1</sup>

Research Institute for Artificial Intelligence, Romanian Academy<sup>2</sup>

{andrei\_marius.avram, george.zaharia0806}@stud.acs.upb.ro

{dumitru.cercel, mihai.dascalu}@upb.ro

## Abstract

Extracting semantic information on measurements and counts is an important topic in terms of analyzing scientific discourses. The 8th task of SemEval-2021: Counts and Measurements (MeasEval) aimed to boost research in this direction by providing a new dataset on which participants train their models to extract meaningful information on measurements from scientific texts. The competition is composed of five subtasks that build on top of each other: (1) quantity span identification, (2) unit extraction from the identified quantities and their value modifier classification, (3) span identification for measured entities and measured properties, (4) qualifier span identification, and (5) relation extraction between the identified quantities, measured entities, measured properties, and qualifiers. We approached these challenges by first identifying the quantities, extracting their units of measurement, classifying them with corresponding modifiers, and afterwards using them to jointly solve the last three subtasks in a multi-turn question answering manner. Our best performing model obtained an overlapping F1-score of 36.91% on the test set.

## 1 Introduction

Our world revolves around quantities and units of measurement present in all texts, ranging from scientific texts to recipes. Nevertheless, the process of automatically extracting measurements is not trivial, considering that, in most situations, the quantitative structures are ambiguous and are not present in the same area within the text. Therefore, parsing the semantic relations becomes a ubiquitous task, since proper quantity identification leads to transformations towards an easy to follow quantitative summary. Advantages of the previously mentioned process can be found in medical prescriptions (Adamo et al., 2015). As such, a system

that can robustly and confidently identify medication quantities, measurement units, as well as the medication itself has the potential to become a breakthrough for computer-based medicine and consultations. Another use case resides in ERP systems where proper parsing of resource descriptions facilitates the identification of similar or duplicate items.

The MeasEval - Counts and Measurements competition (Harper et al., 2021) organized by the 15th International Workshop on Semantic Evaluation (SemEval-2021) creates a new challenge in the area of Natural Language Processing, proposing five subtasks related to span identification, classification, as well as relation extraction, that aim to improve the state of the art for the current field of measurement information extraction. We created a cascaded system to solve the stated problem that is composed of: (1) a subsystem that identifies quantities in the input text; (2) a subsystem that classifies their value modifiers; (3) a subsystem that extracts their measurement unit; and (4) a subsystem that then finds the appropriate measured entities, measured properties, and qualifiers by asking questions related to entity-relations. Three pretrained Transformer-based (Vaswani et al., 2017) language models are experimented for subsystems (1) and (4) of the cascaded system by fine-tuning them on the specific task: Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019), Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019), and Science BERT (SciBERT) (Beltagy et al., 2019). A character-level bidirectional Long Short-Term Memory (BiLSTM) (Hochreiter and Schmidhuber, 1997) architecture was considered for subsystems (2) and (3).

The rest of the paper is structured as follows. The next section presents a series of solutions associated with relation extraction, span identification,

and measurement unit identification. The third section outlines our approaches related to the subtasks proposed by the competition. The fourth section presents a performance evaluation of our systems together with an error analysis, while the final section concludes our work and outlines potential future improvements.

## 2 Related Work

**Span Identification.** Papay et al. (2020) studied the performance of various models designated for different span identification tasks. Out of them, we mention Conditional Random Fields (CRFs) (Lafferty et al., 2001), LSTM cells with CRF, BERT+CRF, LSTM+BERT+CRF, or handcrafted features, usable with any of the previously mentioned models. At the same time, a language model was specifically developed for the span identification tasks, entitled SpanBERT (Joshi et al., 2020), by masking an entire sequence instead of masking a single word in its pretraining process. The authors argued that SpanBERT obtained substantial gains on span selection tasks, such as question answering and coreference resolution.

**Measurement Unit Identification.** Berrahou et al. (2013) proposed a two-step system for search space size reduction, followed by unit extraction from the previously obtained textual fragments. Also, Hundman and Mattmann (2017) presented a hybrid system composed of a CRF that identifies quantities values and their units, followed by a rule-based model to detect their corresponding entities.

**Relation Extraction.** Zhang and Wang (2015) adopted a model based on Recurrent Neural Networks (RNN) (Cho et al., 2014) composed of three main elements: an embedding layer, a bidirectional recurrent layer, followed by a max pooling layer that produces the feature vector used for relation classification. RNN-based models were also applied by Zhang et al. (2015) who adopted BiLSTMs, or by Xiao and Liu (2016) who proposed an architecture based on hierarchical RNNs alongside an attention mechanism. Furthermore, several convolutional neural network-based models with various approaches were proposed, for example: multi-level attention (Wang et al., 2016), attention-based context vectors (Shen and Huang, 2016), or multi-level features (word, lexical, sentence) (Zeng et al., 2014). BiLSTMs are also present in the work of Lee et al. (2019) who implemented a mechanism

based on entity-aware attention using latent entity typing. Jin et al. (2020) approached the relation extraction task by employing a Graph Neural Network system that modeled each relation as a node and learned the dependencies between the nodes.

## 3 Method

Our approach on MeasEval consisted of a cascade system composed of individual subsystems for each of the problems in the first two subtasks, and then jointly solving the last three subtasks with a single subsystem.

### 3.1 Quantity Identification

The subtask of identifying quantities in text was formalized as a sequence labeling problem with Inside–Outside–Beginning (IOB) tags (Ramshaw and Marcus, 1999) that were predicted by a pretrained language model with a CRF on top of predicted logits, as proposed by Avram et al. (2020). The architecture is depicted in Figure 1.

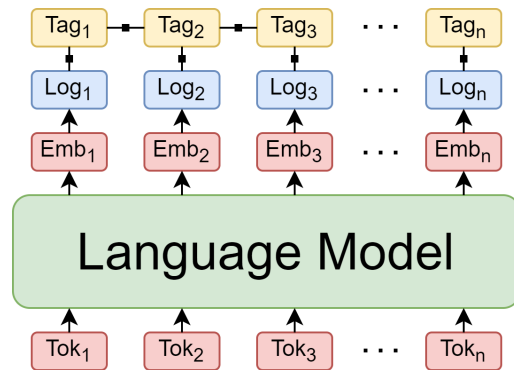


Figure 1: Quantity identification subsystem architecture.

More formally, we project each output embedding  $e_i$  produced by the pretrained language model into probability logits  $l_i$  by using a feed-forward network with a ReLU activation as  $l_i = ReLU(W_l^T e_i + b_l)$ , where  $W_l$  is the corresponding weight matrix and  $b_l$  is the corresponding bias. Then, we model the output conditional probabilities for each tag  $y_i$  by using the CRF learning algorithm, as depicted in Eq. 1:

$$p(y|l) = \frac{1}{Z} \exp \left\{ \sum_{i=1}^n W_{y_{i-1}, y_i}^T l_i + b_{y_{i-1}, y_i} \right\} \quad (1)$$

where  $W_{y_{i-1}, y_i}$  and  $b_{y_{i-1}, y_i}$  are the weight matrix and the bias of the CRF, and  $Z$  is a normalization constant such that the probabilities sum up to one.

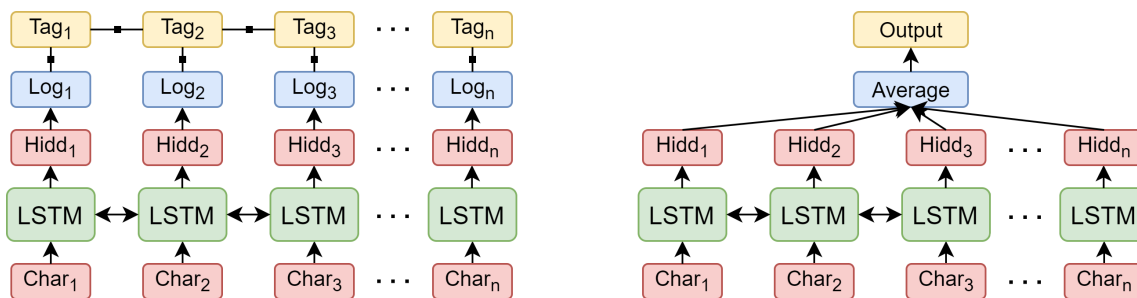


Figure 2: Architectures used in unit extraction (left) and value modifiers classification (right) subsystems.

The entire subsystem is trained to maximizing the log-likelihood of the data, while the Viterbi algorithm (Forney, 1973) is used during inference to find the most likely sequence of tags.

### 3.2 Unit Extraction and Value Modifier Classification

For the second subtask, a character-level BiLSTM extracts the units from quantities and classifies their corresponding value modifiers. We approached the unit extraction in a similar way as the quantity identification, by treating the problem as a sequence tagging; however, the pretrained language model was replaced with BiLSTM cells. Moreover, instead of predicting a label for each character (token), we instead averaged the BiLSTM hidden states and projected their average in an eleven-dimensional vector (i.e., number of possible value modifiers) for the classification. Then, a sigmoid activation function was applied to obtain a vector that contains the probability of the quantity to belong to a class at each index. The architectures used for unit extraction and value modifiers classification are depicted in Figure 2.

### 3.3 Joint Entity Identification and Relations Extraction

**Subtask Grouping.** The last three subtasks were grouped into a single subtask where a pretrained language model was fine-tuned to jointly identify three elements: the span of the measured entities, the measured properties, and their corresponding qualifiers. The model extracts the relations between the three elements and the previously extracted quantities using a multi-turn question answering (QA) architecture, as proposed by Li et al. (2019). The pretrained language models used for this task were identical to the ones from the quantity identification subtask.

**Question Templates.** The input to the subsystem is created by appending a question before the

text that denotes a possible relation between a given and a target entity. There are a total of six question templates that can be filled with the corresponding entities that cover all the possible relations, as depicted in Table 1. Then, the questions are asked in a specific order to correctly identify the relations and the span of the entities. First, starting with a given quantity, the model is asked to identify its measured properties. If a measured property is found, the model marks its span and links it to the quantity with the `HasQuantity` relation (question 1). Second, the model is asked to identify the measured entity with that corresponding measured property, linking the two with the `HasProperty` relation (question 2). Third, if no measured property is found for a given quantity, the model is asked to directly identify the measured entity, marking the relation between the measured entity and the quantity directly with `HasQuantity` (question 3). Finally, once all quantities, measured entities, and properties are identified, the model is asked to identify corresponding qualifiers and marks the relations accordingly (questions 4-6 in table).

**Model Output.** The architecture proposed in (Devlin et al., 2019) for SQuAD 2.0 (Rajpurkar et al., 2018) is employed to create the output of the subtasks; as such, two vectors are used for fine-tuning: a starting vector  $S$  and an ending vector  $E$ . The probability of token  $i$  to be the start of a span is computed as a dot-product between the embedding  $T_i$  and the start vector  $S$ , followed by a softmax applied over all the tokens of the input:  $P = \text{softmax}(T_i \cdot S)$ . An analogous formula computes the end probabilities of a span. Then, we take the indices  $i$  and  $j$  are taken to compute the most probable span for an entity, where  $i \leq j$  maximizes the sum of log-likelihoods  $T_i \cdot S + T_j \cdot E$ . We compare for each query the previously defined maximum sum with the sum of the start and end log-likelihoods of the `[CLS]` token  $s_{null}$  because

#	Relation Type	Question
1	HasQuantity	What is the <i>measured property</i> of the <i>quantity</i> ____?
2	HasProperty	What is the <i>measured entity</i> that has the <i>measured property</i> ____ of the <i>quantity</i> ____?
3	HasQuantity	What is the <i>measured entity</i> that has the <i>quantity</i> ____?
4	Qualifies	What is the <i>qualifier</i> corresponding to the <i>quantity</i> ____?
5	Qualifies	What is the <i>qualifier</i> corresponding to the <i>measured entity</i> ____?
6	Qualifies	What is the <i>qualifier</i> corresponding to the <i>measured property</i> ____?

Table 1: Question templates for each relation type.

there can be questions without an answer<sup>1</sup>. If this sum is higher, then there is no such type of relationship for that entity. A threshold added to the  $s_{null}$  is considered in order to provide a higher granularity between questions with or without answers, which was tuned on the development set to maximize F1-score.

Figure 3 introduces our architecture for entity recognition and relation extraction. The question tokens marked with Qst and the paragraph tokens marked with Tok are fed as input, while the start S and the end E logits are present at output.

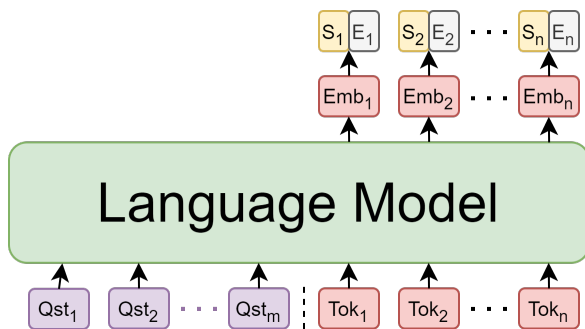


Figure 3: Joint entity recognition and relation extraction architecture as multi-turn question answering.

## 4 Performance Evaluation

### 4.1 Experimental Setup

**Dataset Analysis and Processing.** The provided corpus for the competition was quite scarce, counting 298 samples in both train and trial datasets. The corpus contained texts from the scientific domain, counting a total of 7,979 unique words with an average sentence length of approximately 160 words. For training our models, we merged the train and trial subsets and randomly split them into 90% training and 10% development.

**Pretrained Language Models.** An Adam optimizer (Kingma and Ba, 2014) with a learning rate

<sup>1</sup>Only measured properties and qualifiers related questions are allowed to not have an answer. Measured entity-related questions must always have an answer.

of  $2e-5$  was used for training the subsystem of the first and last three subtasks. We experimented with the large versions of BERT and RoBERTa, and with the base version of SciBERT because there are currently no implementations available online of its large variant. Each subsystem was fine-tuned for 10 epochs; the subsystems that employed large language model variants were trained with a batch size of 2 due to the computational constraints, whereas the subsystems that employed base language model variants used a batch size of 8.

**BiLSTM Networks.** An Adam optimizer was also employed for training the BiLSTM networks of the second subtask, but with a learning rate of  $1e-4$ . The models were trained for 25 epochs using a batch size of 16. We stacked the LSTM cells two times and used a hidden size of 64, with an embedding of 32 dimensions for the characters.

### 4.2 Results

The results of each subsystem on the development set are introduced in Table 3. SciBERT obtained the highest F1-score on both quantity identification and joint entity and relation extraction, although it is smaller when compared with the other two models. On the second subtasks, the model achieved a reasonable performance, 95.75% F1-score on unit extraction and 88.94% F1-score on value modifier classification.

The results of the cascaded system are presented in Table 2 that introduces the global precision, recall, and F1-scores averaged across all subtasks, as well as the exact match (EM) and overlap F1 scores<sup>2</sup> between the gold annotations and our predictions. As opposed to the performance of each subsystem on the development set where SciBERT was the best performing model, RoBERTa obtained the highest scores as a whole system, with an overlap F1-score of 39.05% on the development set and 36.91% on the test set, outperforming SciBERT

<sup>2</sup>The overlap F1-score was the metric by which the competition systems were ranked.

System	Avg. Precision		Avg. Recall		Avg. F1		EM		Overlap F1	
	Dev	Test	Dev	Test	Dev	Test	Dev	Test	Dev	Test
BERT-related	<b>61.10</b>	58.69	52.28	48.05	56.63	52.84	31.61	25.58	36.72	32.69
RoBERTa-related	60.04	<b>61.01</b>	<b>56.05</b>	<b>52.66</b>	<b>58.14</b>	<b>56.53</b>	<b>34.98</b>	<b>30.89</b>	<b>39.05</b>	<b>36.91</b>
SciBERT-related	56.73	54.35	54.61	46.12	55.65	49.90	30.82	23.71	35.89	30.30

Table 2: Results averaged across all five subtasks on the development and test sets.

Subsystem	Precision	Recall	F1
<i>Quantity identification</i>			
RoBERTa-CRF	90.77	92.85	91.26
BERT-CRF	<b>91.77</b>	93.72	92.38
SciBERT-CRF	91.60	<b>95.02</b>	<b>93.00</b>
<i>Unit extraction and value modifier classification</i>			
Unit Extraction	96.44	95.27	95.75
Value Modifiers	91.82	86.65	88.94
<i>Joint relation extraction and entity identification</i>			
RoBERTa-QA	71.04	71.26	71.14
BERT-QA	72.18	<b>71.09</b>	71.63
SciBERT-QA	<b>73.81</b>	70.71	<b>72.22</b>

Table 3: Performance analysis on the development set.

with over 6% and over 3%, respectively. More surprisingly, SciBERT also obtained a lower score than BERT on both sets, having an overlap F1-score lowered by 2% and 1%. We believe that these differences between the scores of RoBERTa and SciBERT were caused by the way the two models were evaluated as stand-alone subsystems or as a whole system.

### 4.3 Error Analysis

**Quantity Sensitivity.** The main drawback in our approach was that all other subtasks were highly dependent on the quality of the extracted quantities for the first subtask. To exemplify this, let us consider the case where a modifier like "approximate" is missed before a quantity; afterwards, it would be impossible to correctly classify its modifiers. Another use case is when the subsystem misses the measuring unit, with the same effect on the unit extractor. Moreover, we noticed that the joint entity and relation extraction was especially sensible to partially identified quantities, producing mostly bad outputs in these cases.

**Measured Unit Inference.** Another limitation of our approach emerges when the unit extractor does not identify all units in a sequence tagging style. This happened in cases when the unit was split across several places in the quantity, or when it had to be predicted from the context. For instance,

the correct unit would be "m<sup>2</sup>" when encountering a "300 m x 400 m" quantity; however, our model found only "m" as unit.

**Long Documents.** Finally, several documents had a longer sequence length than 512 tokens<sup>3</sup> when tokenized, which surpasses the maximum admitted length by the pretrained language models; the workaround was to simply remove the tokens after position 512. However, this solution has the obvious effect of missing identifiable entities that appear after this position.

## 5 Conclusions and Future Work

This paper introduces our approach that solves all the five subtasks of the 8th task of SemEval-2021 competition in a cascaded manner. First, quantities are identified as a sequence tagging task by using a pretrained language model with a CRF layer. Then, the measurement units are extracted and the modifiers are classified using BiLSTMs at character level on the identified quantities. Finally, the measured entities, measured properties, and qualifiers are jointly identified, together with their relations, by using a multi-turn question answering approach with hand-crafted questions specific to each relation type. Our best model obtained an F1-score of 36.91% on the test set. We further emphasized several limitations of our approach and showed that the overall performance was highly sensitive to the quality of the identified quantities.

A possible direction for future work is to test the system using language models that can process longer sequences, such as Longformer (Beltagy et al., 2020) or BigBird (Zaheer et al., 2020), in order to reduce the effect of missing entities simply due to the sequence length. We also consider creating an ensemble model using several pretrained language models to boost the overall performance, as reported by Ionescu et al. (2020).

<sup>3</sup>Approximately 4% of the documents had more than 512 tokens for each pretrained language model.

## References

- Francesco Adamo, Filippo Attivissimo, Attilio Di Niso, and Maurizio Spadavecchia. 2015. An automatic document processing system for medical data extraction. *Measurement*, 61:88–99.
- Andrei-Marius Avram, Dumitru-Clementin Cercel, and Costin Chiru. 2020. UPB at SemEval-2020 task 6: Pretrained language models for definition extraction. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 737–745.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611.
- Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.
- Soumia Lilia Berrahou, Patrice Buche, Juliette Dibia-Barthelemy, and Mathieu Roche. 2013. How to extract unit of measure in scientific documents? In *KDIR: Knowledge Discovery and Information Retrieval*, pages 454–459. Springer.
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Kyle Hundman and Chris A Mattmann. 2017. Measurement context extraction from text: Discovering opportunities and gaps in earth science. *arXiv preprint arXiv:1710.04312*.
- Marius Ionescu, Andrei-Marius Avram, George-Andrei Dima, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. UPB at FinCausal-2020, tasks 1 & 2: Causality analysis in financial documents using pretrained language models. In *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, pages 55–59.
- Zhijing Jin, Yongyi Yang, Xipeng Qiu, and Zheng Zhang. 2020. Relation of the relations: A new paradigm of the relation extraction problem. *arXiv preprint arXiv:2006.03719*.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.
- Jooheon Lee, Sangwoo Seo, and Yong Suk Choi. 2019. Semantic relation classification via bidirectional lstm networks with entity-aware attention using latent entity typing. *Symmetry*, 11(6):785.
- Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1340–1350.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Sean Papay, Roman Klinger, and Sebastian Padó. 2020. Dissecting span identification tasks with performance prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4881–4895.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

- Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer.
- Yatian Shen and Xuan-Jing Huang. 2016. Attention-based convolutional neural network for semantic relation extraction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2526–2536.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Linlin Wang, Zhu Cao, Gerard De Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention cnns. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307.
- Minguang Xiao and Cong Liu. 2016. Semantic relation classification via hierarchical recurrent neural network with attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1254–1263.
- Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*.
- Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*, pages 2335–2344.
- Dongxu Zhang and Dong Wang. 2015. Relation classification via recurrent neural network. *arXiv preprint arXiv:1508.01006*.
- Shu Zhang, Dequan Zheng, Xinchun Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia conference on language, information and computation*, pages 73–78.



# IITK@LCP at SemEval-2021 Task 1: Classification for Lexical Complexity Regression Task

Neil Shirude\*      Sagnik Mukherjee\*

Tushar Shandhilya      Ananta Mukherjee      Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)

{neilrs, sagnikm, anantam, stushar}@iitk.ac.in

ashutoshm@cse.iitk.ac.in

## Abstract

This paper describes our contribution to SemEval 2021 Task 1: Lexical Complexity Prediction. In our approach, we leverage the ELECTRA model and attempt to mirror the data annotation scheme. Although the task is a regression task, we show that we can treat it as an aggregation of several classification and regression models. This somewhat counter-intuitive approach achieved an MAE score of 0.0654 for Sub-Task 1 and MAE of 0.0811 on Sub-Task 2. Additionally, we used the concept of weak supervision signals from Gloss-BERT in our work, and it significantly improved the MAE score in Sub-Task 1.

## 1 Introduction

With the rapid growth in digital pedagogy, English has become an extremely popular language. Although English is considered an easy language to learn and grasp, a person’s choice of words often affects texts’ readability. The use of difficult words can potentially lead to a communication gap, thus hampering language efficiency. Keeping these issues in mind, many Natural Language Processing tasks for text simplification have been recently proposed (Paetzold and Specia, 2017; Sikka and Mago, 2020). Our task of lexical complexity prediction is an important step in the process of simplifying texts.

The SemEval 2021 Task 1 (Shardlow et al., 2021) focuses on lexical complexity prediction in English. Given a sentence and a token from it, we have to predict the complexity score of the token. The task has two Sub-Tasks-

**Sub-Task 1:** complexity prediction of single words

**Sub-Task 2:** complexity prediction of multi word expressions (MWEs).

A word might seem complex because of 2 major

factors-

a) The word is less common or complex in itself.

b) The context in which the word is used makes it hard to comprehend.

Observing the orthogonality of these two reasons, we captured the context-dependent features and independent features separately, trained models on them individually, and then combined the two using ensemble methods. We used the ELECTRA (Clark et al., 2020) model for extracting context-dependent features and GloVe embeddings (Pennington et al., 2014) for representing the word-level features.

Additionally, we propose a classification pipeline that is trained on GloVe embeddings of the tokens. This pipeline can be interpreted as a model for capturing different annotators’ thought processes: overconfidence, under-confidence and randomness. We are making our code available for our models and experiments via GitHub<sup>1</sup>.

## 2 Background

This task uses the CompLex dataset (Shardlow et al., 2020), which is a lexical complexity prediction dataset in English for single and multi word expressions (2-grams). Sentences in this task consists of sentences taken from 3 corpora- Bible, Biomed and Europarl. The train, validation and test split of the data was 9179, 520, 1103 respectively. We used the train data as the validation set.

The aim of the task is to predict how complex a given token in a given sentence is. More mathematically, given a tuple  $[s, t, c]$ , where  $s = [t_1, t_2, \dots, t_n]$  and  $t = t_j$ , we have to give an estimate of the function  $\sigma$ , such that  $\sigma(s, t) = c$ . ( $s$  is the sentence,  $t$  is the token and  $c$  is the complexity score).

The earlier focus on this task has been through

<sup>1</sup><https://github.com/neilrs123/Lexical-Complexity-Prediction>

\* Authors equally contributed to this work.

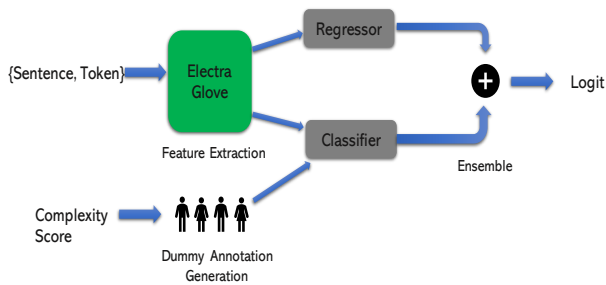


Figure 1: Solution Pipeline

the SemEval 2016 Task 11 (Paetzold and Specia (2016a)). However, it was a binary classification task. Most of the participating systems used Support Vector Machines such as Kuru (2016) and Choubey and Pateria (2016), decision trees and random forests (Choubey and Pateria (2016), Brooke et al. (2016), Ronzano et al. (2016)), and even basic threshold based approaches (Kauchak (2016), Malmasi et al. (2016)). Very few of them, including Bingel et al. (2016) used neural networks. The system by Wróbel (2016) achieved an F1 score very close to the winning solution using only single feature - word frequency from Wikipedia. Most of these systems use word embeddings, POS information and word frequencies as features. The winning system by Paetzold and Specia (2016b) however uses 69 morphological, semantic and syntactic features.

Another related shared task was presented at the BEA workshop at 2018 (Yimam et al., 2018). It had a probabilistic task as well as a binary classification task. Even there, the organizers conclude that feature engineering has worked better than neural networks. The winning system by Gooding and Kochmar (2018) uses feature engineering and later random forest and linear regression models.

### 3 System Overview

Our proposed pipeline can be divided into the following 4 main components-

- a) Feature Extraction
- b) Regression Pipeline
- c) Classification Pipeline
- d) Ensemble

The pipeline is shown in Figure 3.

#### 3.1 Feature Extraction

ELECTRA is a transformer based model, that is trained like a discriminator and not like generator. And in our case, this model performed exception-

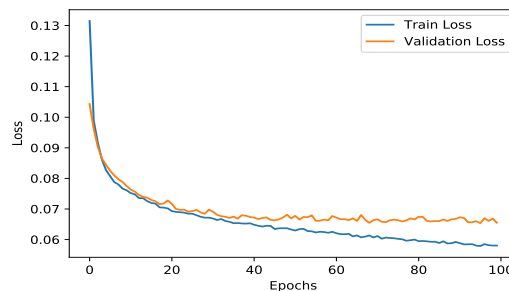


Figure 2: Convergence of losses for finetuning ELECTRA with weak supervision

ally well on the validation data as compared to BERT (Devlin et al., 2019).

We extracted context-dependent features using embeddings generated from the ELECTRA model and captured context-independent word-level features using static 200-dimensional GloVe embeddings of the tokens.

In order to generate the embeddings of the target word through ELECTRA, we implemented the KMP pattern matching algorithm (Wikipedia, 2021) to find the indices of the sub-tokens of the target token in the tokenized sentence. Subsequently, we calculated an average across these sub-token embeddings generated by ELECTRA.

While using GloVe embeddings, in the case of multi-word expressions in Sub-Task 2, the average of the embeddings of both token words was taken as the feature vector. If a word was not present in the GloVe dictionary, the GloVe embedding was initialized to a 200-dimensional vector consisting of zeros.

#### 3.2 Regression Pipeline

The most natural way to look at the lexical complexity prediction task is to treat it as a regression task. The regression pipeline, a significant component of our system, is based on this idea. For Sub-task 1, in the regression pipeline, a pretrained ELECTRA model was finetuned with a linear layer on top of it. We leveraged the model directly available at the Huggingface library (Wolf et al., 2020). Only the last transformer layer of ELECTRA was kept trainable. The remaining ones were kept frozen. For Sub-task 2, a fixed ELECTRA model (non-trainable weights) was used to generate token embeddings and a linear regression model was trained with these extracted embeddings.

**Weak Supervision:** In order to have higher attention on the target word, the use of weak supervi-

sion signals proved useful. Inspired by GlossBert (Huang et al., 2019), the target word was wrapped with single inverted commas ( ' )s as a weak signal to the transformer (Vaswani et al., 2017) model. This technique significantly improved the results obtained using the regression pipeline in subtask I. However, the same technique applied to subtask II made the scores worse.

Method	Val MAE	Test MAE
+ signal	0.06516	0.06800
- signal	0.06990	0.07118

Table 1: Variation of MAE scores with and without the signalling technique for Sub-task 1: the single word task. ('+ signal' means weak supervision has been used and '- signal' means otherwise.)

### 3.3 Classification Pipeline

**Motivation from Annotation Procedure:** Another way to look at the task is via a novel classification pipeline that is inspired from the data annotation process that is explained in Shardlow et al. (2020). Even though the task is a regression task, each data annotator performed a 5 class classification-

Given a sentence and a token in the sentence, each annotator had to select one class from among Very Easy, Easy, Neutral, Difficult and Very Difficult. Each of these classes was mapped to a discrete label between 0 and 1- namely 0, 0.25, 0.5, 0.75 and 1 respectively. The final complexity score was an average of up to 20 such annotations.

The Classification Pipeline aims to model this data annotation procedure. The main idea of this process is to teach classification models how to annotate data tuples. The three main components of this scheme are-

**a)** Generating dummy annotations from complexity scores **b)** Training classification models on dummy annotations, and **c)** Aggregating all predicted annotations to generate predicted complexity scores.

**Generation of Dummy Annotations:** A given complexity score can be represented as a weighted average of its lower and upper target classes and the weights can be determined using the magnitude of the complexity score. These weights then determine the proportions of the two classes in the set of dummy annotations for that data tuple. For example, if the number of dummy annotators is  $n = 5$  and the complexity score of the training

example is  $c = 0.2$ , the lower and upper target classes are  $low = 0$  and  $high = 0.25$ , respectively. Let  $\alpha$  be the proportion of dummy annotations with the lower target class. Correspondingly,  $1 - \alpha$  will be the proportion with the upper target class. The number of dummy annotations with  $target\_class = low$  are given as  $floor(n * \alpha)$  and that with  $target\_class = high$  as  $n - floor(n * \alpha)$ .  $\alpha$  can be calculated using the equation-

$$c = \alpha * low + (1 - \alpha) * high$$

We get  $\alpha = 0.2$ . Hence, we have  $floor(n * \alpha) = 1$  dummy annotations with  $target\_class = low(0)$  and remaining 4 annotations with  $target\_class = high(0.25)$ . Hence, the dummy annotations set for  $c = 0.2$  is 0, 0.25, 0.25, 0.25, 0.25. Similarly, the dummy annotations set for  $c = 0.8$  is 0.75, 0.75, 0.75, 0.75, 1.

In this process, we also attempted to capture the impact of intentional human errors made during the data annotation procedure. Just like a weary or uninterested annotator who would have randomly selected for one of the five classes for a certain data tuple, a small fraction of the dummy annotations was assigned random values from the set containing 0, 0.25, 0.5, 0.75 and 1. This modification aims to model the small-scale randomness in annotation procedure.

Using this procedure, dummy annotation sets of size  $n$  can be generated for any value of  $c$ , where  $n$  can be treated as a hyperparameter. The value  $n$  can also be interpreted as the number of classification models that are being trained in the next step.

**Classification Models:** In a diverse set of annotators, there will be over-confident annotators who will select lower classes and there will be under-confident annotators who will select upper classes. Then there will be neutral annotators as well. By ensuring that the dummy annotations are sorted, we can say that the first classifier learns how to annotate like the over-confident annotator, the last classifier learns how to annotate like the under-confident annotator and the classifiers in between model the neutral annotators. We trained SVM classifiers with RBF kernels, using GloVe embeddings of token words as features.

**Aggregation of Predicted Annotations:** The annotations were aggregated by simply taking the average of all predicted class labels in order to obtain the final predicted complexity scores. Each of these models may have high individual variances,

Complexity Score	Dummy Annotations				
0.2	1	2	2	2	2
0.8	4	4	4	4	5
0.35	2	2	2	3	3
0.4	2	2	3	3	3

1st  
classifier

2nd  
classifier

3rd  
classifier

4th  
classifier

5th  
classifier

Figure 3: A few worked out examples of generating dummy annotations from complexity scores. For each of these cases, the continuous labels 0,0.25,0.50,0.75 and 1 are mapped to categorical labels 1,2,3,4,5 and then put into SVM. Clearly the labels of the 1st classifier is less than that of the second one. i.e. on a scale of confidence, the first classifier is at a lesser position. So it models a less confident person.

but the ensemble tends to have lower variance and bias. Also, any number of models can be inserted in the ensemble without leading to over-fitting on the train data.

### 3.4 Ensemble

In order to have a better bias variance trade off and also to exploit the “expertise” of different pipelines, the final approach incorporates both the regression and classification pipelines to form an ensemble. The final predicted complexity was obtained by taking an ensemble of the predictions from the regression and classification pipelines as described above. The classification pipeline for both the Sub-Tasks was based on GloVe embeddings as features and SVM classifiers. The regression pipeline for Sub-Task 1 was based on fine-tuning ELECTRA with weak supervision and that for Sub-Task 2 was based on features collected from ELECTRA model (non-trainable) with a linear regression trained on it.

## 4 Experimental Setup

The official evaluation metric for both the Sub-Tasks was Pearson Correlation (standard for regression tasks). For both sub-tasks, the train/test/val split as per the official release has been used. The ELECTRA finetuning was done with an NVIDIA GTX 1080 GPU with early stopping (93 epochs). We used the MAE loss function to train the model with an adam optimizer with  $lr = 1e^{-5}$ ,  $eps = 1e^{-8}$  and  $weightdecay = 0$ . Training set was shuffled and the batch size was kept at 64. In the ELECTRA model, the padding parameter was set

to True and maximum length was at 140. For the SVM models the value of slack was chosen to be 1 and for SVM and Linear regression the sklearn (Pedregosa et al., 2011) library was used. All the hyperparameters were tuned with a grid search method.

## 5 Results

**Results on Validation Data:** The comparison of the baseline results and our results obtained using the regression pipeline, the classification pipeline and the ensemble of the two models on the validation set (trial data) is given in Table 4.

Task	Baseline	Regression Pipeline	Classification Pipeline
Subtask 1	0.0853	0.0651	0.0641
Subtask 2	-	0.0840	0.0768

Table 2: Results on validation set (Mean Absolute Errors)

Task	MAE	Pearson	MSE
One	0.0623	0.8308	0.0065
Two	0.0727	0.8146	0.0087

Table 3: Results on Validation Set for final ensemble

**Results on Test Data:** Our results on the test data along with the best results obtained for each task are shown in Table 1.

The winning system’s pearson and MAE scores on the test data are as follows: 0.7886 and 0.0609 for subtask I(single word expressions), 0.8612 and 0.0616 for subtask II(multi word expressions).

Task	MAE	Pearson	MSE
One	0.0654	0.7511	0.0071
Two	0.0811	0.8277	0.0098

Table 4: Results on Test Set

## 6 Error Analysis

Analyzing all the experiments and the corresponding results, the following can be concluded: **a)** Word-level features as well as context-dependent features need to be considered while determining

complexity of a token. **b)** Approaches based on the data annotation scheme are well suited to tackle the lexical complexity prediction task. **c)** Ensemble of a large number of simple models is an effective way of tackling this task. **d)** Models with large number of parameters like BERT () suffer heavily due to overfitting, where as ELECTRA base prove to be much better.

The model architectures that were tried out in earlier stages showed similar trends. For example, ELECTRA finetuning produced much better scores than BERT finetuning. Also, simpler models like a simple linear regression on GloVe embeddings showed promise, proving that simpler models with lesser parameters worked better. All these trends across those models are visually shown in Figure 4. It was observed that the model was underperforming on the tuples from Biomed corpus. However the scores did not improve using BERT variants like BioBERT (Lee et al., 2019), BioMedBERT (Chakraborty et al., 2020) and a few other transformer based models pretrained on biomedical texts. A variant of ELECTRA on biomedical texts could have improve on this, however due to its unavailability it could not be tried out.

In majority of the prior work on LCP, there is abundance use of word frequency as a feature. However, in this system the scores got worse when frequency features were used along with others in ensemble. And the feature in itself could not produce competitive results. Previously, Gong et al. (2020) and Mu et al. (2018) have shown that frequency information causes significant distortion in the embedding space. We also hypothesize that the frequency information in GloVe embeddings help us in this regard.

## 7 Conclusion

In this paper we presented a system for lexical complexity prediction in the form of a regression task. The proposed system’s primary novelty is in treating it as a classification task and trying to model the annotation scheme. An ensemble of these classification models and vanilla fine-tuning of ELECTRA model proved to be very useful. Also the weak supervision based approach gave the scores a significant boost for the Sub-Task 1.

## References

Joachim Bingel, Natalie Schluter, and Héctor Martínez Alonso. 2016. *CoastalCPH at SemEval-*

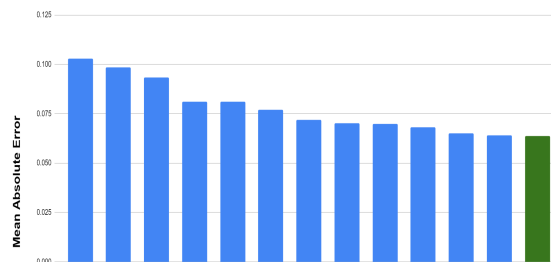


Figure 4: Comparison of MAE values of the models we tried (Subtask I). From left the models are (1) Linear Regression with hand crafted features, (2) Character level RNN, (3) Character level CNN, (4) and (5) sentence and character level GRU and LSTMs (6), (7), (8), (9) and (10) Linear Regression with GloVe, ELECTRA and BERT embeddings, (11) the current regression pipeline, (12) classification pipeline, (13) Final ensemble

2016 task 11: The importance of designing your neural networks right. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1028–1033, San Diego, California. Association for Computational Linguistics.

Julian Brooke, Alexandra Uitdenbogerd, and Timothy Baldwin. 2016. *Melbourne at SemEval 2016 task 11: Classifying type-level word complexity using random forests with corpus and word list features.* In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 975–981, San Diego, California. Association for Computational Linguistics.

Souradip Chakraborty, Ekaba Bisong, Shweta Bhatt, Thomas Wagner, Riley Elliott, and Francesco Mosconi. 2020. *BioMedBERT: A pre-trained biomedical language model for QA and IR.* In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 669–679, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Prafulla Choubey and Shubham Pateria. 2016. *Garuda & Bhasha at SemEval-2016 task 11: Complex word identification using aggregated learning models.* In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1006–1010, San Diego, California. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. *Electra: Pre-training text encoders as discriminators rather than generators.*

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of deep bidirectional transformers for language understanding.*

- Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2020. [Frage: Frequency-agnostic word representation](#).
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. [GlossBERT: BERT for word sense disambiguation with gloss knowledge](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.
- David Kauchak. 2016. [Pomona at SemEval-2016 task 11: Predicting word complexity based on corpus frequency](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1047–1051, San Diego, California. Association for Computational Linguistics.
- Onur Kuru. 2016. [AI-KU at SemEval-2016 task 11: Word embeddings and substring features for complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1042–1046, San Diego, California. Association for Computational Linguistics.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. [Biobert: a pre-trained biomedical language representation model for biomedical text mining](#). *Bioinformatics*.
- Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. [LTG at SemEval-2016 task 11: Complex word identification with classifier ensembles](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000, San Diego, California. Association for Computational Linguistics.
- Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2018. [All-but-the-top: Simple and effective postprocessing for word representations](#).
- Gustavo Paetzold and Lucia Specia. 2016a. [SemEval 2016 task 11: Complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016b. [SV000gg at SemEval-2016 task 11: Heavy gauge complex word identification with system voting](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974, San Diego, California. Association for Computational Linguistics.
- Gustavo H. Paetzold and Lucia Specia. 2017. A survey on lexical simplification. *J. Artif. Int. Res.*, 60(1):549–593.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. [Scikit-learn: Machine learning in python](#). *Journal of Machine Learning Research*, 12(85):2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Francesco Ronzano, Ahmed Abura'ed, Luis Espinosa-Anke, and Horacio Saggion. 2016. [TALN at SemEval-2016 task 11: Modelling complex words by contextual, lexical and semantic features](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1011–1016, San Diego, California. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [Semeval-2021 task 1: Lexical complexity prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Punardeep Sikka and Vijay Mago. 2020. [A survey on text simplification](#).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).
- Wikipedia. 2021. [Knuth–morris–pratt algorithm — Wikipedia, the free encyclopedia](#). [https://en.wikipedia.org/w/index.php?title=Knuth%E2%80%93pratt\\_algorithm&oldid=1005503556](https://en.wikipedia.org/w/index.php?title=Knuth%E2%80%93pratt_algorithm&oldid=1005503556). [Online; accessed 17-February-2021].
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen,

Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Huggingface’s transformers: State-of-the-art natural language processing](#).

Krzysztof Wróbel. 2016. [PLUJAGH at SemEval-2016 task 11: Simple system for complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 953–957, San Diego, California. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

# LCP-RIT at SemEval-2021 Task 1: Exploring Linguistic Features for Lexical Complexity Prediction

Abhinandan Desai, Kai North, Marcos Zampieri, Christopher Homan

Rochester Institute of Technology

Rochester, NY, USA

ad2724@rit.edu, kn1473@rit.edu, mazgla@rit.edu, cmh@cs.rit.edu

## Abstract

This paper describes team LCP-RIT’s submission to the SemEval-2021 Task 1: Lexical Complexity Prediction (LCP). The task organizers provided participants with an augmented version of CompLex (Shardlow et al., 2020), an English multi-domain dataset in which words in context were annotated with respect to their complexity using a five point Likert scale. Our system uses logistic regression and a wide range of linguistic features (e.g. psycholinguistic features,  $n$ -grams, word frequency, POS tags) to predict the complexity of single words in this dataset. We analyze the impact of different linguistic features on the classification performance and we evaluate the results in terms of mean absolute error, mean squared error, Pearson correlation, and Spearman correlation.

## 1 Introduction

Lexical complexity prediction (LCP) is the task of predicting the complexity value of a target word within a given text (Shardlow et al., 2020). Complexity within LCP is used as a “synonym for difficulty” (Malmasi and Zampieri, 2016)<sup>1</sup>. A complex word is therefore a word that a target population may find difficult to understand. Various LCP systems have been designed to identify words that may be found to be complex for children (Kajiwara et al., 2013), language learners (Malmasi et al., 2016), or people suffering from a reading disability, such as dyslexia (Rello et al., 2013). These systems have been utilized within assistive language technologies, lexical simplification systems, and in a variety of other applications.

LCP is related to complex word identification (CWI) (Paetzold and Specia, 2016). CWI is modeled as a binary classification task by assigning each target word with a complex or non-complex

<sup>1</sup>The term “complex” within LCP is not necessarily related to the terms *simplex* and *complex* used in morphology.

label. The shortcomings of modeling lexical complexity using binary labels have been discussed in previous work (Zampieri et al., 2017; Maddela and Xu, 2018), motivating the organization of SemEval-2021 Task 1: Lexical Complexity Prediction.<sup>2</sup> LCP models complexity in a continuum and the goal is to predict a target word’s degree of complexity by assigning it a value between 0 and 1. This value may then correspond to one of the following labels: very easy (0), easy (0-0.25), neutral (0.25-0.5), difficult (0.5-0.75), or very difficult (0.75-1) (Shardlow et al., 2020).

In this paper, we describe (in detail in Section 4) the LCP-RIT entry to SemEval-2021 Task 1. We approached LCP from a feature engineering perspective with a particular focus on the adoption of psycholinguistic features, such as average age-of-acquisition (AoA), familiarity, prevalence, concreteness, and arousal, alongside the use of prior complexity labels. Our submitted system utilized a combination of these linguistic features, which we compared to a baseline model that only used statistical features: word length, word frequency and syllable count (Quijada and Medero, 2016; Mukherjee et al., 2016). On our training dataset, our submitted system achieved a mean absolute error (MAE) of 0.067, mean squared error (MSE) of 0.007, Person Correlation (R) score of 0.779, and a Spearman Correlation ( $\rho$ ) score of 0.724. This surpassed our baseline model’s performance by a MAE of 0.008, MSE of 0.003, as well as R and  $\rho$  scores of 0.075 and 0.062 respectively.

## 2 Related Work

Before SemEval-2021 Task 1: LCP, two CWI shared tasks were organized at one SemEval-2016 and the other at BEA-2018 (Paetzold and Specia, 2016; Yimam et al., 2018). While the first CWI provided participants with an English dataset, the

<sup>2</sup><https://sites.google.com/view/lcpsharedtask2021/home>



second provided a multilingual dataset. The systems submitted to the English track of the second shared task (Yimam et al., 2018) performed better overall than the previous task (Paetzold and Specia, 2016), probably due to the properties of the two datasets (Zampieri et al., 2017). State-of-the-art neural net models and word embedding models performed worse than conventional models such as decision trees (DTs) and random forests (RFs) (Yimam et al., 2018). Among the conventional models, the use of statistical, character  $n$ -gram, and psycholinguistic features was found to be highly effective in improving CWI performance (Malmasi et al., 2016; Zampieri et al., 2016; Paetzold and Specia, 2016; Yimam et al., 2018).

Among the best performing systems in CWI 2018, Gooding and Kochmar (2018) used an ensemble of classifiers. They found that during their system’s development, the boosting classifier AdaBoost, a random forest classifier, or a combination of both classifiers achieved the highest performance. These systems used multiple features such as the word’s grammatical category, Google character  $n$ -gram frequency as well as a range of psycholinguistic features (Gooding and Kochmar, 2018).

Of the remaining systems, Aroyehun et al. (2018) and Hartmann and Borges dos Santos (2018) utilized statistical features, such as word length and number of syllables, psycholinguistic features such as familiarity, age of acquisition, concreteness, and imagery scores, and word  $n$ -grams. Hartmann and Borges dos Santos (2018) compared the performance of tree ensembles to a convolutional neural network (CNN). They found that their tree ensembles performed better than their CNN, especially when the target expression contained more than three words (Aroyehun et al., 2018).

### 3 Task and Dataset

The LCP shared task organizers provided participants with the CompLex corpus, an English multi-domain dataset with sentences from the Bible, the European Parliament proceedings, and a collection of biomedical texts. A pool of annotators, using a five point Likert scale, labeled the complexity of single words and multi-word expressions in CompLex (Shardlow et al., 2020).

Taking advantage of the annotation of single words and multi-word expressions, the LCP shared task was divided into two sub-tasks as follows:

- **Sub-task 1:** predicting the complexity score for single words;
- **Sub-task 2:** predicting the complexity score for multi-word expressions.

We chose to participate in sub-task 1. Sub-Task 1’s training dataset contained 7,662 instances with its test dataset having 917 instances. 20% of the training dataset was used to test our system’s performance during development. Sub-Task 1 received 54 system submissions.

## 4 System Overview

### 4.1 Model

Taking inspiration from the CWI systems discussed in Section 2, we adopted a random forest regressor (RFR) to predict the complexity values of each word within the test dataset. To achieve this, we tested the impact a variety of linguistic features have on LCP performance during our system’s development. The RFR was taken from scikit-learn’s ensemble module (Pedregosa et al., 2011). The RFR used a maximum of 120 trees and 750 features.

### 4.2 Features

We constructed a baseline RFR using the following statistical features and character trigrams. We then used psycholinguistic and additional features to see whether its baseline performance could be improved.

**Statistical Features** include word length, word frequency and syllable count. Zipf’s Law implies that words that appear less frequently within a text are likely to be longer and therefore may be considered more complex than words that are more frequent and shorter (Quijada and Medero, 2016). In addition, words with a high number of syllables are difficult to pronounce and are subsequently hard to read (Mukherjee et al., 2016). As such, word length, word frequency and syllable count were considered to be good baseline statistical indicators of a word’s complexity value.

**Character N-grams** include the use of character bigram and trigram frequencies. These frequencies were calculated by counting each bigram’s and trigram’s presence in the target words provided in Sub-Task 1’s training dataset. Experimentation with bigrams and trigrams, along with a combination of both, found that the use of trigrams on their own was superior. This together with their use

within prior CWI systems justified their inclusion within our baseline model (Yimam et al., 2018).

**Psycholinguistic Features** include average age of acquisition (AoA), concreteness, familiarity, prevalence and arousal. AoA is the age at which a word’s meaning is first learned. Concreteness refers to “the degree to which the concept denoted by a word refers to a perceptible entity” (Brysbaert et al., 2013). Familiarity and prevalence are somewhat similar. Familiarity is how well known the word is to an individual and was obtained through self-report (Gilhooly and Logie, 1980). Prevalence was calculated in accordance to the percentage of people who knew the word (Brysbaert et al., 2019). Lastly, arousal is a measure of how active or passive a word’s meaning is interpreted as being<sup>3</sup>. For instance, the word “*nervous*” indicates more arousal than “*lazy*” (Mohammad, 2018). As such, grammatical categories such as adjectives, verbs, and adverbs may incite higher levels of arousal than nouns.

Average AoA was calculated by averaging the AoAs provided in the Living Word Vocabulary Dataset (Dale and O’Rourke, 1981) with an updated version of this dataset (Brysbaert and Biemiller, 2017). Both datasets consisted of AoA values for 44,000 English word meanings. Concreteness, familiarity and arousal values were taken from the MRC Psycholinguistic Database (Wilson, 1988) as well as three newer datasets each containing 37,058, 61,858 and 20,000 English words (Brysbaert et al., 2013, 2019; Mohammad, 2018).

**Additional Features** include part-of-speech (POS) tags as well as prior complexity labels. POS tags were generated by using the Python Natural Language Toolkit (Bird et al., 2009). Prior complexity labels were taken from the previous CWI shared tasks (Paetzold and Specia, 2016; Yimam et al., 2018) and the Word Complexity Lexicon (Maddela and Xu, 2018). A combined dataset was then created that contained a total of 26,088 English words each with a binary complexity value.

## 5 Evaluation

### 5.1 Features

To determine the effect each feature had on our baseline model’s performance, we used the fol-

<sup>3</sup>The terms “active” and “passive” do not refer to the use of *active* or *passive voice* but rather the emotional or physical intensity associated with a word’s meaning (Mohammad, 2018).

lowing scores: mean absolute error (MAE), mean squared error (MSE), Pearson Correlation (R) and Spearman Correlation ( $\rho$ ). Table 1 depicts each feature’s performance on the training dataset. These criteria have also been used in the SemEval LCP test set evaluation.

Average AoA decreased the baseline model’s MAE and MSE by 0.004 and 0.001 respectively. It likewise increased its R and  $\rho$  scores by 0.039. This generated the second highest R and  $\rho$  scores of 0.743 and 0.701 respectively. Average AoA is therefore a useful feature for LCP.

Brysbaert et al.’s prevalence and concreteness (Brysbaert et al., 2013, 2019) were also seen to improve the baseline model’s performance with prevalence being the most notable. Prevalence decreased baseline MAE and MSE scores by 0.005 and 0.002 respectively. It also surpassed baseline R by 0.054 and  $\rho$  by 0.047, yielding the highest increases among all features. Concreteness (Brysbaert et al., 2013) also caused a slight increase in the baseline model’s scores, being greater than that caused by MRC concreteness. Concreteness values (Brysbaert et al., 2013) increased the baseline model’s R and  $\rho$  scores by 0.032 and 0.024 respectively, whereas the MRC concreteness values resulted in a slightly less impressive increase of 0.019 in both its R and  $\rho$  scores. However, there was little-to-no difference in MAE and MSE produced by either set of concreteness values.

Features	Performance			
	R	$\rho$	MAE	MSE
Baseline Features	0.704	0.662	0.075	0.010
Average AoAs	0.743	0.701	0.071	0.009
<b>Prevalence</b>	<b>0.758</b>	<b>0.709</b>	<b>0.070</b>	<b>0.008</b>
MRC Familiarity	0.727	0.687	0.073	0.009
Concreteness	0.736	0.686	0.072	0.009
MRC Concreteness	0.723	0.681	0.073	0.009
Arousal	0.722	0.676	0.074	0.009
POS Tags	0.701	0.663	0.075	0.010
Complexity Labels	0.727	0.686	0.072	0.009

Table 1: Feature performance on training dataset. The baseline model uses the statistical features and character trigrams. Best results in bold.

Two possible conclusions can be drawn: 1). The difference in the calculation of prevalence versus that of familiarity likely causes prevalence to be a greater indicator of word complexity<sup>4</sup>, and 2). The

<sup>4</sup>Prevalence being the percentage of people who. know the word (Brysbaert et al., 2019). Familiarity being a self-reported

superior coverage of Brysbaert et al.’s prevalence (2018) and concreteness (2013) datasets (Brysbaert et al., 2019, 2013): being 52.62% and 57.51% respectively, compared to that of the MRC Psycholinguistic Database (Wilson, 1988): being 23.44%, suggests that there now exists larger and more up-to-date psycholinguistic datasets that are more useful for LCP feature engineering.

Arousal has never before been used for LCP or CWI. Due to its ability to differentiate grammatical categories, such as nouns and verbs, along with its ability to signify a word’s intensity, we had speculated that arousal would be able to help predict a word’s complexity. Arousal was found to have no significant effect on the baseline model’s performance. Nevertheless, once added to our submitted system, it slightly decreased its MSE by 0.001 and increased its R and  $\rho$  scores by 0.002 and 0.001 respectively.

POS tags was the worst performing feature as POS tags had little affect on improving our model’s performance. It achieved the same MAE and MSE values as our baseline model: 0.075 and 0.01 respectively. Regarding R score, only a slight increase of 0.001 was observed. POS tags was the only feature that saw a decrease in our model’s  $\rho$  score, worsening its performance by 0.003. This suggests that a word’s grammatical category may not impact its degree of complexity. This is also supported by Arousal’s lack of improved performance.

Given that prior complexity labels are directly related to complexity prediction, it was believed that they would be the most influential in improving overall performance. Instead, AoA, prevalence and concreteness were all found to be more beneficial with higher or identical MAE, MSE,  $\rho$ , and R scores. Complexity labels only saw a slight decrease in MAE and MSE by 0.003 and 0.001 respectively and a slight increase in  $\rho$  and R scores by 0.023 and 0.024 respectively. The binary nature of prior CWI datasets is likely responsible for this phenomenon, as binary 0 or 1 complexity values are not well suited for a regression-based task, such as LCP. This would have resulted in the same problem faced by previous CWI systems: the misclassification of words on the decision boundary.

measure of an individual’s awareness of the word (Gilhooly and Logie, 1980).

## 5.2 Models

The results for the three models on the training dataset are presented in Table 2. This is then followed by a short description of each model as well as our performance on the test dataset.

Model	Performance			
	R	$\rho$	MAE	MSE
Model 1	0.772	0.717	0.068	0.008
Model 2	0.777	0.724	0.067	0.008
LCP-RIT	<b>0.779</b>	<b>0.724</b>	<b>0.067</b>	<b>0.007</b>

Table 2: Model performance on training dataset.

**Model 1 - Top 3 Features:** Adding the top 3 features of average AoA, prevalence and concreteness to our baseline model reduced its MAE and MSE by 0.007 and 0.002 respectively and increased its R score by 0.068 and its  $\rho$  score by 0.055. It attained a new MAE of 0.068 which was noticeably better than our baseline model’s previous MAE of 0.075. This goes to show that inclusion of psycholinguistic features has a positive impact on the performance of an LCP system.

**Model 2 - Top 5 Features:** A small improvement was seen after adding the fourth and fifth best performing features to Model 1, namely, MRC familiarity and prior complexity labels. Model 2 increased Model 1’s R and  $\rho$  scores by 0.005 and 0.007. However, it failed to improve Model 1’s MAE or MSE. This small increase in performance was due to the prior top 3 features of average AoA, prevalence and concreteness already having captured those instances caught by MRC familiarity and prior complexity labels. This further proves the redundancy of the MRC Psycholinguistic Database (Wilson, 1988) as well as binary complexity labels for LCP feature engineering.

**LCP-RIT:** Our final model submitted to the official evaluation used the psycholinguistic features of average AoA, prevalence, concreteness and arousal together with our baseline model’s features of word length, syllable count, word frequency and character trigrams to predict the lexical complexity of single words. On the training dataset of SemEval-2021 Task 1: LCP, we achieved a MAE of 0.067, MSE of 0.007, R score of 0.779, and  $\rho$  score 0.724. We performed less well on the single word test dataset with an MAE and MSE of 0.072 and 0.009 respectively and  $\rho$  and R scores of 0.709 and 0.653 respectively. This reduced performance may be

indicative of our submitted system being overfit on our training dataset.

## 6 Conclusion

We carried out multiple experiments evaluating the impact of linguistics features in LCP using the CompLex dataset for English. We have shown that several psycholinguistic features help with LCP. Average AoA, prevalence and concreteness were all found to be beneficial, whereas MRC familiarity, MRC concreteness and prior complexity labels were proven to be redundant. We would like to explore other features described in [Shardlow et al. \(2021\)](#). In terms of performance, we believe that the multiple features we tested allowed us to get close to the maximum performance for this dataset using regression. A possible alternative for better performance is to test state-of-the-art transformer models. Furthermore, we are interested in looking at the performance of these features for LCP in languages other than English and for multilingual datasets.

## Acknowledgments

We would like to thank the LCP shared task organizers for proposing this interesting shared task and for making the data available.

## References

- Segun Taofeek Aroyehun, Jason Angel, Daniel Alejandro Pérez Alvarez, and Alexander Gelbukh. 2018. Complex Word Identification: Convolutional Neural Network vs. Feature Engineering. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.
- Marc Brysbaert and Andrew Biemiller. 2017. Test-based age-of-acquisition norms for 44 thousand english word meanings. *Behavioural Research*, 49:1520–1523.
- Marc Brysbaert, Pawel Mandera, Samantha McCormick, and Emmanuel Keuleers. 2019. Word prevalence norms for 62,000 english lemmas. *Behavior Research Methods*, 51:467–479.
- Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2013. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior Research Methods*, 46:904–911.
- Edgar Dale and Joseph O'Rourke. 1981. *The living word vocabulary, the words we know: A national vocabulary inventory*. World Book.
- Ken Gilhooly and Robert Logie. 1980. Age-of-acquisition, imagery, concreteness, familiarity, and ambiguity measures for 1,944 words. *Behavior Research Methods & Instrumentation*, 12(4):395–427.
- Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI Shared Task 2018: Complex Word Identification with Ensemble-Based Voting. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Nathan Hartmann and Leandro Borges dos Santos. 2018. NILC at CWI 2018: Exploring Feature Engineering and Feature Learning. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*, pages 59–73.
- Mounica Maddela and Wei Xu. 2018. A word-complexity lexicon and a neural readability ranking model for lexical simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3749–3760.
- Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. LTG at SemEval-2016 Task 11: Complex Word Identification with Classifier Ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.
- Shervin Malmasi and Marcos Zampieri. 2016. MAZA at SemEval-2016 Task 11: Detecting Lexical Complexity Using a Decision Stump Meta-Classifer. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.
- Saif Mohammad. 2018. Obtaining Reliable Human Ratings of Valence, Arousal, and Dominance for 20,000 English Words. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, Melbourne, Australia. Association for Computational Linguistics.
- Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. JU\_NLP at SemEval-2016 Task 11: Identifying Complex Words

- in a Sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.
- Febian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Oliver Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Maury Quijada and Julie Medero. 2016. HMC at SemEval-2016 Task 11: Identifying Complex Words Using Depth-limited Decision Trees. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.
- Luz Rello, Ricardo Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. 2013. Frequent words improve readability and short words improve understandability for people with dyslexia. In *Proceedings of the INTERACT 2013: 14th IFIP TC13 Conference on Human-Computer Interaction. Cape Town, South Africa, 2013*. INTERACT.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.
- Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior research methods, instruments, & computers*, 20(1):6–10.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Luci Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*, New Orleans, United States. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in Data Annotation and System Performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. MacSaar at SemEval-2016 Task 11: Zipfian and Character Features for Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, San Diego, California. Association for Computational Linguistics.

# Alejandro Mosquera at SemEval-2021 Task 1: Exploring Sentence and Word Features for Lexical Complexity Prediction

Alejandro Mosquera  
Symantec Enterprise Division  
Broadcom Corporation  
alejandro.mosquera@broadcom.com

## Abstract

This paper revisits feature engineering approaches for predicting the complexity level of English words in a particular context using regression techniques. Our best submission to the Lexical Complexity Prediction (LCP) shared task was ranked 3rd out of 48 systems for sub-task 1 and achieved Pearson correlation coefficients of 0.779 and 0.809 for single words and multi-word expressions respectively. The conclusion is that a combination of lexical, contextual and semantic features can still produce strong baselines when compared against human judgement.

## 1 Introduction

Lexical complexity is a factor usually linked to poor reading comprehension (Dubay, 2004) and the development of language barriers for target reader groups such as second language learners (Saquete et al., 2013) or native speakers with low literacy levels, effectively making texts less accessible (Rello et al., 2013). For this reason, complex word identification (CWI) is often an important sub-task in several human language technologies such as text simplification (Siddharthan, 2004) or readability assessment (Collins-Thompson, 2014).

The Lexical Complexity Prediction (LCP) shared task of Semeval-2021 (Shardlow et al., 2021) proposes the evaluation of CWI systems by predicting the complexity value of English words in context. LCP is divided into two sub-tasks: Sub-task 1, predicting the complexity score for single words; Sub-task 2: predicting the complexity score for multi-word expressions. In our participation in both sub-tasks, we treat the identification of complex words as a regression problem, where each word is given a score between 1 and 5, given the sentence in which it occurs. In order to do this, we have evaluated sub-sets of word and sentence features against different machine learning models.

Our best submissions achieved Pearson correlation coefficients of 0.779 and 0.809 for single words and multi-word expressions respectively.

In Section 2 we review related work for this task. Section 3 and 4 introduce the data and feature engineering approaches respectively. In Section 5 the performance of different machine learning models is analysed. In Section 6 we present the obtained results. Finally, in Section 7 we draw our conclusions and outline future work.

## 2 Related Work

Previous CWI studies applied to the English language have relied mostly on word frequencies, psycholinguistic information (Devlin and Tait, 1998), lexicons and other word-based features such as number of characters or syllable counts (Shardlow, 2013), which considered in most cases the target word in isolation. In order to address the limitations of word-level approaches more recent work made use of contextual and sentence information such as measuring the complexity of word n-grams (Ligozat et al., 2012), applying language models (Maddela and Xu, 2018) or treating the problem as a sequence labelling task (Gooding and Kochmar, 2019).

In this paper, we not only evaluate many of the traditional word-based features found in the literature but we also pay attention to the context surrounding the target by generating additional bigram and sentence features. In the end, we demonstrate that a careful selection of simple features is still competitive against more novel approaches for this task.

## 3 Datasets

CompLex (Shardlow et al., 2020), which was the official dataset provided by the organizers, contains complexity annotations using a 5-point Likert scale

for 7,662 words and 1,517 multi-word expressions (MWE) from three domains: the Bible, Europarl, and biomedical texts.

External datasets and models are historically allowed and used in SemEval as a way of complementing the original training set. Likewise, based on previous experiences, external resources can also correlate better with the evaluation labels than the official task resources in certain scenarios (Mosquera, 2020). For this reason, related datasets from previous CWI shared tasks such as CWI 2016 (Paetzold and Specia, 2016) and CWI 2018 (Štajner et al., 2018) were considered and evaluated as both extra training data and for deriving additional features. However, the performance of our models during the validation step not only didn't improve but worsened when attempting to use them.

## 4 Feature Engineering

The 51 features used in order to detect the complexity of single words and each component of MWEs are as follows:

**Word length** (word\_len): The length in characters of the target word.

**Syllable count** (syl\_count): Target word syllable count.

**Morpheme length** (morpheme\_len): Number of morphemes for the target word.

**Google frequency** (google\_freq): The frequency of the target word based on a subset Google ngram corpus<sup>1</sup>.

**Wikipedia word frequency** (wiki\_freq1): The frequency of the target word based on Wikipedia<sup>2</sup>.

**Wikipedia document frequency** (wiki\_freq2): The number of documents in Wikipedia where the target word appears.

**Complexity score** (comp\_lex): Complexity score for the target word from a complexity lexicon (Maddela and Xu, 2018).

**Number of morphemes** (morpheme\_len): Number of morphemes in the target word.

**Zipf frequency** (zip\_freq): The frequency of the target word in Zipf-scale as provided by the wordfreq (Speer et al., 2018) Python library.

**Kucera-Francis word** (kucera\_francis): Kucera-Francis (Kucera et al., 1967) frequency of the target word.

**Kucera-Francis lemma** (st\_kucera\_francis): Kucera-Francis (Kucera et al., 1967) frequency of the target word lemma.

**Is stopword** (stop): True if the target word is an stopword.

**Is acronym** (acro): Heuristic that is set to True if the target word is a potential acronym based on simple casing rules.

**Average age of acquisition** (age): At what age the target word is most likely to enter someone's vocabulary (Kuperman et al., 2012).

**Average concreteness** (concrete): Concretedness rating for the target word (Brysbaert et al., 2014).

**Lemma length** (lemma\_len): Lemma length of the target word.

**Word frequency (COCA)** (word\_freq): Frequency of the target word based on the COCA corpus (Davies, 2008).

**Lemma frequency (COCA)** (lemma\_freq): Frequency of the lemmatized target word based on the COCA corpus (Davies, 2008).

(consonant\_freq): Frequency of consonants in the target word.

**Number word senses** (wn\_senses): Number of senses of the target word extracted from WordNet (Fellbaum, 2010).

**Number of synonyms** (synonyms): Number of synonyms of the target word from WordNet.

**Number of hypernyms** (hypernyms): Number of hypernyms of the target word from WordNet.

**Number of hyponyms** (hyponyms): Number of hyponyms of the target word from WordNet.

**WordNet min-depth** (wn\_mindepth): Minimum distance to the root hypernym in WordNet for the target word.

**WordNet max-depth** (wn\_maxdepth): Maximum distance to the root hypernym in WordNet for the target word.

**Number of Greek or Latin affixes** (greek\_or\_latin\_affix): True if the target word contains Greek or Latin affixes<sup>3</sup>.

**Bing frequency** (bing\_counts): The frequency of the target word based on the Bing n-gram corpus (Wang et al., 2010).

**Bi-gram frequency** (ph\_mc2): Bi-gram frequency for the target and its preceding word in Google Books Ngram Dataset obtained via the phrasefinder API<sup>4</sup>.

<sup>1</sup><https://github.com/hackerb9/gwordlist>

<sup>2</sup>[https://github.com/alex-pro-dev/english-words-by-frequency/blob/master/wikipedia\\_words.zip](https://github.com/alex-pro-dev/english-words-by-frequency/blob/master/wikipedia_words.zip)

<sup>3</sup><https://github.com/sheffieldnlp/cwi>

<sup>4</sup><https://phrasefinder.io/api>

**Volume count** (ph\_vc2): The number of books where the target and its preceding word appeared in the Google Books Ngram Dataset obtained via the phrasefinder API.

**Year of appearance** (ph\_fy2): The first year where the target and its preceding word appeared in the Google Books Ngram Dataset obtained via the phrasefinder API.

**SUBTLEX-US frequency** (FREQcount): Target word frequency based on SUBTLEX-US corpus (Brysaert et al., 2012).

**SUBTLEX-US number of films** (CDcount): Number of films in which the target word appears in the SUBTLEX-US corpus.

**SUBTLEX-US frequency lowercase** (FREQlow): Number of times the target word appears in the SUBTLEX-US corpus starting with a lowercase letter

**SUBTLEX-US number of films lowercase** (Cdlow): Number of films in which the target word appears starting with a lower-case letter in the SUBTLEX-US corpus.

**SUBTLEX-US frequency per million** (SUBTLWF): Target word frequency per million words in the SUBTLEX-US corpus.

**SUBTLEX-US log frequency** (Lg10WF): The base-10 logarithm of the absolute frequency of the target word plus one in the SUBTLEX-US corpus.

**SUBTLEX-US percentage of films** (SUBTLCD): The percentage of the films where the target word appears in the SUBTLEX-US corpus.

**SUBTLEX-US log number of films** (Lg10CD): The base-10 logarithm of the number of films in which the target word appears in the SUBTLEX-US corpus.

**SUBTLEX-UK frequency** (LogFreqZipf): The base-10 logarithm of the target word frequency in Zipf-scale for the SUBTLEX-UK corpus (Van Heuven et al., 2014).

**SUBTLEX-UK Cbeebies frequency** (LogFreqCbeebiesZipf): The base-10 logarithm of the target word frequency in Zipf-scale for the Cbeebies subset of SUBTLEX-UK corpus

**SUBTLEX-UK CBBC frequency** (LogFreqCBBCZipf): The base-10 logarithm of the target word frequency in Zipf-scale for the CBBC subset of SUBTLEX-UK corpus

**SUBTLEX-UK BNC frequency** (LogFreqBNCZipf): The base-10 logarithm of the target word frequency in Zipf-scale for the BNC subset of SUBTLEX-UK corpus

**ANC word frequency** (anc): Frequency of the target word based on the American National Corpus (ANC) (Ide and Macleod, 2001).

**Kincaid grade level** (sentence\_Kincaid): Kincaid grade level of the whole sentence.

**ARI score** (sentence\_ARI): Automated readability index (Senter and Smith, 1967) of the whole sentence.

**Coleman-Liau score** (sentence\_Coleman-Liau): Coleman-Liau readability score (Coleman and Liau, 1975) of the whole sentence.

**Flesch score** (sentence\_FleschReadingEase): Flesch reading ease score (Flesch, 1948) of the whole sentence.

**Gunning-Fog** (sentence\_GunningFogIndex): Gunning-Fog readability index (Gunning et al., 1952) of the whole sentence.

**LIX score** (sentence\_LIX): LIX readability score (Anderson, 1983) of the whole sentence.

**SMOG index** (sentence\_SMOGIndex): SMOG readability index (Mc Laughlin, 1969) of the whole sentence.

**RIX score** (sentence\_RIX): RIX readability score (Anderson, 1983) of the whole sentence.

**Dale-Chall index** (sentence\_DaleChallIndex): Dale-Chall readability index (Chall and Dale, 1995) of the whole sentence.

All the readability features were calculated using the readability Python library <sup>5</sup>.

## 5 Machine Learning Approach

Since the labels in the training dataset were continuous we have modelled both sub-tasks as regression problems. For sub-task 1, we made use of LightGBM (LGB) (Ke et al., 2017) implementation of gradient tree boosting. Minimal hyper-parameter optimization was performed against our development set, using a 0.01 learning rate and limiting the number of leaves of each tree to 30 over 500 boosting iterations.

For sub-task 2, the complexity score of each MWE component was obtained by using a linear regression (LR) model and averaged with equal weights.

By examining the feature importance for both the LGB model in Figure 2 and the LR model in Figure 3 we can observe several sentence readability features being identified as top contributors. While some degree of correlation between the complexity of the sentence and the target word was expect

<sup>5</sup><https://github.com/andreasvc/readability>



a priori, a machine learning model can also use sentence-level complexity as a predictor of formality and genre (Mosquera and Moreda, 2011), thus being able to differentiate between the different sub-corpora present in the training data as seen in Figure 1.

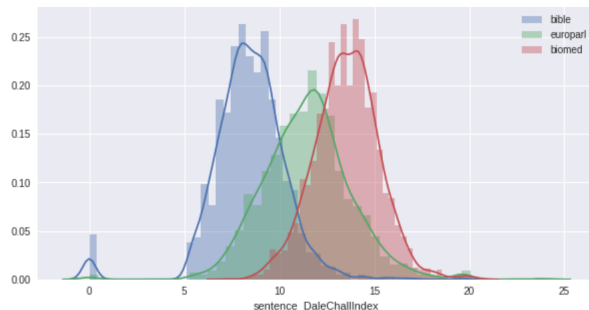


Figure 1: Example of differences in readability across the CompLex sub-corpora as measured by the Dale-Chall index.

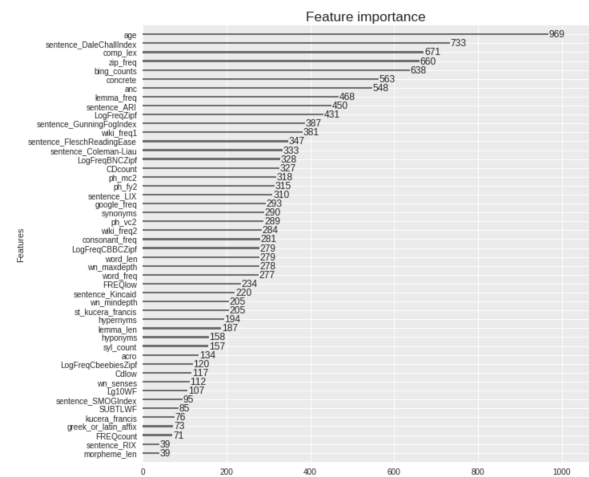


Figure 2: LGB feature importance as the number of times the feature is used in the model.

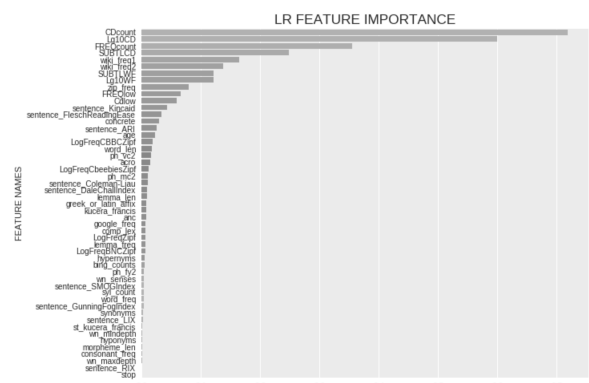


Figure 3: Linear regression weights.

Model	Dev	Trial	Test
Sub-task 1 LR	0.789	0.798	0.760
Sub-task 1 RF	0.792	0.824	0.766
Sub-task 1 LGB	0.801	0.841	<b>0.779</b>
Sub-task 2 LR	0.771	0.780	<b>0.809</b>
Sub-task 1 winner			<b>0.788</b>
Sub-task 2 winner			<b>0.861</b>

Table 1: Performance comparison of different models for each sub-task and evaluation set using Pearson’s r.

## 6 Results

For sub-task 1, we have evaluated the performance of both linear and tree ensembles using the provided trial set and a randomly selected holdout with 30% of the training data as development set. The best performing model was gradient boosting. See Table 1.

## 7 Conclusion and Future Work

In this paper, we present the system developed for the Lexical Complexity Prediction task of SemEval 2021. Even though most of the features we made use of are relatively common in previous works, we demonstrate that a careful selection of lexical, contextual and semantic features at both target word and sentence level can still produce competitive results for this task. In a future work we would like to explore different neural network architectures and automated machine learning (AutoML) approaches.

## References

Jonathan Anderson. 1983. Lix and rix: Variations on a little-known readability index. *Journal of Reading*, 26(6):490–496.

Marc Brysbaert, Boris New, and Emmanuel Keuleers. 2012. Adding part-of-speech information to the sublex-us word frequencies. *Behavior research methods*, 44(4):991–997.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Jeanne Sternlicht Chall and Edgar Dale. 1995. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books.

Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.

- Keryn Collins-Thompson. 2014. [Computational assessment of text readability: A survey of current and future research](#). *ITL - International Journal of Applied Linguistics*, 165:97–135.
- Mark Davies. 2008. The corpus of contemporary american english: 450 million words, 1990-present.
- Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic Databases*, pages 161–173.
- William H. Dubay. 2004. *The principles of readability*. Costa Mesa, CA: Impact Information.
- Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.
- Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.
- Sian Gooding and Ekaterina Kochmar. 2019. [Complex word identification as a sequence labelling task](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153, Florence, Italy. Association for Computational Linguistics.
- Robert Gunning et al. 1952. *Technique of clear writing*.
- Nancy Ide and Catherine Macleod. 2001. The american national corpus: A standardized resource of american english. In *Proceedings of corpus linguistics*, volume 3, pages 1–7. Citeseer.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. [Lightgbm: A highly efficient gradient boosting decision tree](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.
- Henry Kucera, Henry Kučera, and Winthrop Nelson Francis. 1967. *Computational analysis of present-day American English*. University Press of New England.
- Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior research methods*, 44(4):978–990.
- Anne-Laure Ligozat, Cyril Grouin, Anne Garcia-Fernandez, and Delphine Bernhard. 2012. [ANNLOR: A naïve notation-system for lexical outputs ranking](#). In *\*SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 487–492, Montréal, Canada. Association for Computational Linguistics.
- Mounica Maddela and Wei Xu. 2018. [A word-complexity lexicon and a neural readability ranking model for lexical simplification](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3749–3760, Brussels, Belgium. Association for Computational Linguistics.
- G. Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading*, 12(8):639–646.
- Alejandro Mosquera. 2020. [Amsqr at SemEval-2020 task 12: Offensive language detection using neural networks and anti-adversarial features](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1898–1905, Barcelona (online). International Committee for Computational Linguistics.
- Alejandro Mosquera and Paloma Moreda. 2011. [The use of metrics for measuring informality levels in web 2.0 texts](#). In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Luz Rello, Ricardo Baeza-Yates, and Horacio Saggion. 2013. [The impact of lexical simplification by verbal paraphrases for people with and without dyslexia](#). In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2, CICLing'13*, page 501–512, Berlin, Heidelberg. Springer-Verlag.
- Estela Saquete, Sonia Vazquez, Elena Lloret, Fernando Llopis, Jose M. Gomez-Soriano, and Alejandro Mosquera. 2013. Improving reading comprehension of educational texts by simplification of language barriers. In *EDULEARN13 Proceedings*, 5th International Conference on Education and New Learning Technologies, pages 3784–3792. IATED.
- R.J. Senter and Edgar A. Smith. 1967. Automated readability index. Technical report, CINCINNATI UNIV OH.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [Complex: A new corpus for lexical complexity prediction from likert scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.

- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Advait Siddharthan. 2004. [Syntactic simplification and text cohesion](#). *Research on Language & Computation*, 4.
- Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. [Luminosinsight/wordfreq: v2.2](#).
- Walter J.B. Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: A new and improved word frequency database for british english. *Quarterly journal of experimental psychology*, 67(6):1176–1190.
- Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.
- Kuansan Wang, Chris Thrasher, Evelyne Viegas, Xiaolong(Shiao-Long) Li, and Bo-June (Paul) Hsu. 2010. [An overview of microsoft web n-gram corpus and applications](#).

# CompNA at SemEval-2021 Task 1: Prediction of lexical complexity analyzing heterogeneous features

**Giuseppe Vettigli**

Centrica plc

giuseppe.vettigli@centrica.com

**Antonio Sorgente**

Institute of Applied Sciences

and Intelligent Systems

National Research Council

antonio.sorgente@isasi.cnr.it

## Abstract

This paper describes the CompNa model that has been submitted to the Lexical Complexity Prediction (LCP) shared task hosted at SemEval 2021 (Task 1). The solution is based on combining features of different nature through an ensembling method based on Decision Trees and trained using Gradient Boosting. We discuss the results of the model and highlight the features with more predictive capabilities.

## 1 Introduction

Complex Word Identification (CWI) is a task focused on the detection of complex (not easy to understand) words or expressions. One of the challenges of natural language-based systems is to provide informative content that is tailored to the needs of users in terms of content and level of understanding. For this aim, predicting Lexical Complexity plays a crucial role in simplifying the text so that it can be more easily understood by people with low literacy levels (for example children) or non native speakers (Shardlow, 2013). The interest of the Computational Linguistics community on this topic has grown in recent years. Indeed, SemEval 2016 presented a challenge specifically on CWI (Paetzold and Specia, 2016a; Zampieri et al., 2017). The task proposed to build models capable of predicting whether a word was easy or not for non-native English speakers. In this case, the proposed dataset was annotated with a binary label.

In 2018 the CWI task was re-proposed at the workshop for Building Educational Applications (BEA) (Yimam et al., 2018). The data was again annotated with binary labels but samples from more languages were considered, specifically English, German and Spanish. In addition to the word classification task, a sub-task where the participants had to predict the probability of a word being complex was added.

The best performing models for both tasks were based on ensembling techniques using features that were carefully selected (Paetzold and Specia, 2016b; Gooding and Kochmar, 2018).

In this context, the Lexical Complexity Prediction (LCP) shared task hosted at SemEval 2021 proposes a challenge where the data is annotated according to the degree of complexity (Shardlow et al., 2021). This type of annotation allows regression models that predict a complexity index rather than just a binary label. This task is split into two sub-tasks, the first one is about the prediction of the complexity of single words and the second is about the prediction of the complexity of multi word expressions.

In this work we present our submission to both the sub-tasks with a model that aggregates a large set of heterogeneous features that can capture a wide variety of linguistics aspects (morphological, semantic, distributional and lexicon-based) in a regression model based on Gradient Boosting. We also present an in-depth analysis of which features are more important for the model.

In Section 2 we present a description of the task and the data available. In Section 3 we introduce the model. In Section 4 we show an analysis of the most relevant features for the model. In Section 5 we present the results. Finally, in Section 6 we draw some conclusions.

## 2 Task and data

The data released for sub-task 1 is made of 9000 sentences where the complexity of a single word was annotated considering its context. The complexity has been annotated using a 5-point Likert scale (from 1 to 5 corresponding to Very Easy, Easy, Neutral, Difficult and Very Difficult) with the values scaled in the range 0-1. Here is one example of a sentence annotated in the dataset:

“The structural Gh gene itself and Stat5b are excellent candidates.”

In this sentence, the word “candidates” is annotated with a complexity of 0.11 (the average in the dataset is 0.30). The goal of sub-task 1 is to predict the complexity of a single word given the sentence and the word to evaluate. The data for sub-task 2 is made of 1800 sentences where expressions of two words are annotated as already described. Here is one sample extracted from the dataset:

“I invite the President to ask all Members who are in the pension fund to say so orally, in plenary, immediately, because they have a direct interest in what is to be discussed.”

In this sentence, the expression “direct interest” is annotated with a complexity of 0.40 (the average in the dataset is 0.42). The goal of sub-task 2 is to predict the complexity of an expression like the one above given the sentence and the expression to evaluate.

For both sub-tasks, the data was selected from three different corpora:

- The World English Bible, translation from Christodouloupoulos and Steedman (2015).
- A selected portion of the European Parliament proceedings in English.
- Selected articles from the biomedical domain.

Having data from sources so diverse is a unique aspect of this task. This dataset was introduced in (Shardlow et al., 2020).

### 3 CompNA model

The main idea behind our model is to aggregate many diverse features that can capture a wide variety of linguistics aspects in a regression model that offers the ability to interpret which of them are more influential.

#### 3.1 Features

In this section we list the features used for the two sub-tasks.

##### 3.1.1 Features for sub-task 1

The following sets of features were used to capture morphological aspects of the text:

- Part of speech tag and Syntactic dependency of the word to evaluate and surrounding words in a window of three words. The library

Spacy with the model `en_core_web_sm` (Honnibal and Montani, 2017) has been used to extract these features.

- Syllables in the word. Number of syllables in the word. Minimum, maximum and average number of syllables in the sentence.
- Length of the word. Minimum, maximum, and average of the lengths of all the words in the sentence.
- Desinence of the word, first and last letter of the word.
- Number of unique characters.
- A Boolean value that is 1 only if all the characters in the word are uppercase.

To take into account distributional characteristic of the word to evaluate we used:

- GloVe embedding of the word, we used the version pre-trained on Wikipedia 2014 and Gigaword 5 with size 50 (Pennington et al., 2014).
- Frequency of the word and of the part of the sentence of three words that include the word in the wordfreqs dataset (Speer et al., 2018).

Semantic aspects were encoded considering the number of synsets and hyponyms of the word in WordNet.

We also considered a set of binary features that report if the word is in one of the following lexicons: Obscure words (Chrisomalis, 1996), Medical words, Simple English words (Ogden and Halász, 1935).

##### 3.1.2 Features for sub-task 2

For sub-task 2 we have used the same features considered for sub-task 1 computing them for each word in the expression to evaluate. We have also restricted the part of speech tags to the target words and added the frequency of the entire expression.

#### 3.2 Regressor

Our final regressor is composed of 120 Decision Trees with a maximum depth of 5 layers. The model was trained using Gradient Boosting (Friedman, 2001) with a learning rate of 0.047 and taking a sub-sample of 75% of the original data at each boosting iteration. The final prediction is computed averaging the output of each tree. The library XGBoost (Chen and Guestrin, 2016) was used for our experiments. The parameters were selected via a grid search performed using the library Scikit-Learn (Pedregosa et al., 2011).

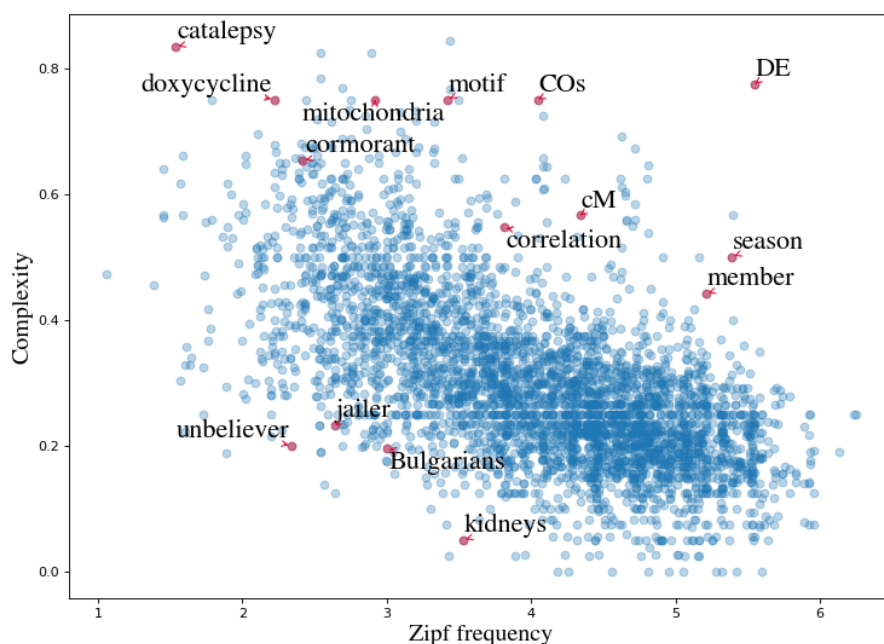


Figure 1: Frequency of the word in Zipf scale versus the complexity. The points in red represent words for which the model has an error higher than a quarter of a point. The chart was made splitting the train data for sub-task 1 in half. The first half has been used to train the model and the second to evaluate the error.

Features group	Importance	N. features
Syllables	19.1%	2260
GloVe	17.8%	50
Desinences	10.2%	898
All frequencies	8.7%	6
Starting letters	5.8%	52

Table 1: Importance of the top 5 groups of features for the model (on single words).

#### 4 Features importance

In order to bring insights into how the model works, we study the contribution of the features. Tables 1 and 2 report the importance of the features that are influential for the prediction, respectively, per group and single feature. The feature importance reflects the number of times a feature is selected for a split for one of the decision trees in the model, weighted by the improvement as a result of the split, and averaged over all the trees in the model (Elith et al., 2008).

Looking at the importance of the features grouped by type we find morphological and distributional ones at the top. While the morphological

Feature	Importance	Type
All uppercase letters	9.47%	binary
Word frequency	7.93%	float
In medical lexicon	3.03%	binary
POS PROP	1.85%	binary
Word length	1.47%	integer
Syllable “ca”	1.19%	binary
Ends with “s”	1.10%	binary
Desinence “ess”	0.84%	binary
Ends with “e”	0.75%	binary
Desinence “ing”	0.71%	binary
GloVe 32th position	0.68%	float
Syllable “ver”	0.67%	binary
Syllable “ro”	0.64%	binary
Desinence “ium”	0.63%	binary
GloVe 29th position	0.62%	float

Table 2: Importance of the top 15 single features for the model (on single words).

group (syllables, desinences and starting letters) add up to almost 3000 features, the distributional group, made of GloVe and word frequencies, only presents 56 features.

Going into details and inspecting the importance

Sub-task	Team	Pearson	Spearman	MAE	MSE	R2
1	JUST Blue	0.7886	0.7369	0.0609	0.0061	0.6172
1	CompNa	0.7552	0.7153	0.0641	0.0070	0.5701
1	<i>baseline</i>	0.6920	0.6533	0.0737	0.0091	0.4387
2	DeepBlueAI	0.8612	0.8526	0.0616	0.0063	0.7389
2	CompNa	0.7931	0.7800	0.0783	0.0093	0.6160
2	<i>baseline</i>	0.7503	0.7435	0.0848	0.0111	0.5386

Table 3: Results of the CompNA model compared a baseline and the best performing model in the competition. The baseline is given by the values on the 20th percentile of the leaderboard.

of single features we note that two of them stand out covering more than 15% of importance. The first is the binary variable representing if the word to evaluate is made of only capital letters and the second is the frequency of the word. In Figure 1 we see the frequency of the word in Zipf scale versus the complexity annotated. The frequency of a word in Zipf scale is the base-10 logarithm of the number of times it appears per billion words (Van Heuven et al., 2014). The Figure shows in red the words for which the model achieves an error greater than 0.25. It is easy to see that the relation between the two features in the chart is well covered by the model as significant errors happen mostly in samples that can be considered outliers. Specifically, we see that many acronyms tend to be outliers in this space. It is interesting to note that acronyms highlighted in the chart are annotated with a score much higher than the average.

Looking again at the table of individual features importance, we find the desinence “ess” which is associated with simple words (such as “darkness”, “kindness” and “business”) and the desinence “ium” which is associated with words that have a complexity more than the average (such as “epithelium”, “cadmium”, “medium”).

Considering the most important syllables, we notice the syllable “ca” which is associated with words from a wide range of complexities (the most complex “cause”, and the least complex “catalepsy”). And the syllable “ver” which is associated with low complexity words (the most complex being “reversal”, and the least complex being “river”).

This analysis further confirms the hypothesis outlined in (Zampieri et al., 2016) that distributional and morphological aspects of the words have a tight bond with the complexity.

The analysis also highlights one of the weak points of the model, it overlooks features related to

the context of the word.

## 5 Results

In Table 3 we summarize the results comparing our model with the best performing model in the competition and a baseline given by the values on the 20th percentile of the leaderboard. The proposed model largely outperforms the baseline in all the measures considered. Notice that the baseline considered here is more challenging than the one proposed in the description of the data (Shardlow et al., 2020). Regarding sub-task 1, the results of the proposed model are comparable to the ones of the winner of the competition in all the measures apart from the  $R^2$  where the two models have a difference of almost 18%. While for sub-task 2 the model beats the baseline but it is far from the winning model. For sub-task 1 the model ranked 26th achieving above average performances and for sub-task 2 it ranked 24th achieving average performances.

## 6 Conclusions

In this paper, we presented a solution to predict the complexity of single and multi word expressions combining a large number of features from a diverse nature.

The model achieves average results and, more importantly, offers the ability to quantify the relevance of the features for the prediction. Thanks to this ability we have shown that the frequency of occurrence and the morphology of the words are key predictors of complexity for the data considered.

From our analysis, it is clear that our model overlooks features related to the context of the word and we would like to improve it under this point of view in our future efforts.

## References

- Tianqi Chen and Carlos Guestrin. 2016. **XGBoost: A scalable tree boosting system**. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, pages 785–794, New York, NY, USA. ACM.
- Stephen Chrisomalis. 1996. The phrontistery: Obscure words and vocabulary resources. Available: <http://phrontistery.info/>.
- Jane Elith, John R Leathwick, and Trevor Hastie. 2008. A working guide to boosted regression trees. *Journal of Animal Ecology*, 77(4):802–813.
- Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- Sian Gooding and Ekaterina Kochmar. 2018. Camb at cwi shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- Charles Kay Ogden and Gyula Halász. 1935. *Basic English*. Kegan Paul Trench Trubner. Available: <http://www.basic-english.org/download/download.html>.
- Gustavo Paetzold and Lucia Specia. 2016a. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Gustavo Paetzold and Lucia Specia. 2016b. Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Marcos Zampieri, and Michael Cooper. 2020. Complex—a new corpus for lexical complexity prediction from likertscale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62.
- Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. **Luminosight/wordfreq: v2.2**.
- Walter JB Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: A new and improved word frequency database for british english. *Quarterly journal of experimental psychology*, 67(6):1176–1190.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*.
- Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. Macsaar at semeval-2016 task 11: Zipfian and character features for complexword identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1001–1005.



# PolyU CBS-Comp at SemEval-2021 Task 1: Lexical Complexity Prediction (LCP)

**Rong Xiang**

Department of Computing  
The Hong Kong Polytechnic University  
csrxiang@comp.polyu.edu.hk

**Jinghang Gu**

Chinese and Bilingual Studies  
The Hong Kong Polytechnic University  
jinghang.gu@polyu.edu.hk

**Emmanuele Chersoni**

Chinese and Bilingual Studies  
The Hong Kong Polytechnic University  
emmanuelechersoni@gmail.com

**Wenjie Li**

Department of Computing  
The Hong Kong Polytechnic University  
cswjli@comp.polyu.edu.hk

**Qin Lu**

Department of Computing  
The Hong Kong Polytechnic University  
csluqin@comp.polyu.edu.hk

**Chu-Ren Huang**

Chinese and Bilingual Studies  
The Hong Kong Polytechnic University  
churen.huang@polyu.edu.hk

## Abstract

In this contribution, we describe the system presented by the PolyU CBS-Comp Team at the Task 1 of SemEval 2021, where the goal was the estimation of the complexity of words in a given sentence context.

Our top system, based on a combination of lexical, syntactic, word embeddings and Transformers-derived features and on a Gradient Boosting Regressor, achieves a top correlation score of 0.754 on the subtask 1 for single words and 0.659 on the subtask 2 for multiword expressions.

## 1 Introduction

The notion of *complexity* has often been debated in linguistics and, depending from the disciplines, it might have different meanings.

In linguistic typology, for example, complexity is generally studied as a property of the language system as a whole, it is conceived as the number of (morphological, syntactic, semantic etc.) distinctions that a speaker has to master, and it is assessed by comparing different languages (McWhorter, 2001; Parkvall, 2008). On the other hand, in the perspective of psycholinguistics and cognitive science, the notion of complexity can be described as the *difficulty* encountered by language users while processing concrete linguistic realizations (sentences, utterances etc.) (Blache, 2011; Chersoni et al., 2016, 2017, 2021; Iavarone et al., 2021; Sarti et al., 2021). Finally, in the Computational Linguistics community, the assessment of complexity at the lexical level is often related to readability

applications (Shardlow et al., 2020), with the goal of determining if a word in a given text will be difficult to understand for the language users. Such applications are extremely useful for second language learners, for speakers with relatively low literacy and for people with reading disabilities, helping to tailor the difficult level of the texts to the needs of the target users.

Task 1 of SemEval 2021 (Shardlow et al., 2021) aims at the development of systems for the estimation of lexical complexity in context, both for single words and for multiword expressions. The organizers provided two datasets with the target words in a sentence context, with annotations consisting of a mean of the complexity ratings assigned by humans. In our paper, we present the system developed by the PolyU CBS-Comp team for the competition. Our top system achieves a Pearson correlation of, respectively, 0.754 on the single words dataset and 0.659 on the multiword expressions one.

## 2 Related Work

In the earliest shared task on the lexical complexity problem, organized in 2016 (Paetzold and Specia, 2016), complexity was defined as a binary variable: given a word in context, the word will be judged as complex or not. Of course, this was a simplifying assumption, since there might be many situations where the boundary is not a clear-cut one, and annotators would rather indicate a value in a continuous scale. Moreover, the "complex" words in the data only needed to be categorized as such by just one

of the annotators. A further study by [Zampieri et al. \(2017\)](#) analyzed the output of the participating systems, showing that modeling complexity as binary actually hindered their performance.

A second iteration of the shared task was organized in 2018 ([Yimam et al., 2018](#)), this time features two separate subtasks: the traditional binary classification task, where systems had to predict whether one word was complex or not, and a regression task, where systems had to estimate the probability that an annotator would have considered a given word as complex.

Recently, [Shardlow et al. \(2020\)](#) have introduced CompLex, a new gold standard for the estimation of lexical complexity in context for English: the corpus, including sentences from different textual genres, is annotated with the mean complexity ratings for the target words. As a preliminary evaluation, the authors presented the results of a linear regression model trained on sets of features including word and sentence embeddings and some hand-crafted features that are traditionally associated to complexity, such as frequency, word length and syllable count. The best scores, in terms of mean absolute error, were obtained when using only the latter set of features, while models based on the dimensions of the embeddings were lagging behind.

### 3 Datasets

The datasets for the shared task are part of the CompLex corpus, which has been published and described by [Shardlow et al. \(2020\)](#). The annotated sentences were collected using three different corpora: the Europarl corpus ([Koehn, 2005](#)), which includes the proceedings of the European Parliament; the CRAFT biomedical corpus ([Bada et al., 2012](#)); and the Bible, in the modern version of the World English Bible translation ([Christodouloupoulos and Steedman, 2015](#)).

The organizers selected targets as either single words (Sub-Task 1) or multiword expressions (Sub-Task 2), and the datasets include also multiple examples with the same target, as different contexts can determine different complexity values. As for the multiword expressions, they were identified via syntactic patterns, being either adjective-noun or noun-noun phrases.

20 annotations per data instance were collected, with annotators coming from different English-speaking countries (US, UK and Australia): the possible ratings ranged from 1 → Very Easy to 5

Dataset Instance	Corpus	Score
This was the <b>length</b> of Sarah’s life.	Bible	0.125
... <b>dissenters</b> by definition excluded.	Europarl	0.688
...due to reduction in <b>adipose</b> tissue...	CRAFT	0.813

Table 1: Examples of the instances from the different corpora, together with the mean complexity scores for the target words in bold.

→ Very Difficult. Mean scores were then normalized in the 0-1 range.

In a first phase, the organizers released a training data of 7661 samples for the single words track and 1517 samples for the multiword expressions track, together with a trial/validation dataset of 420 and 99 samples, respectively. Later, they released a test set of 917 samples for the single words track and 184 samples for the multiword expressions track.

Examples of the instances are shown in Table 1.

## 4 Evaluation

For both the single words and the multiword expressions track, we used the same set of features as input for a regression algorithm. In the multiword expressions track, we computed the value of the features for each of the two words in the target expression and then we took the average.

### 4.1 Features

As hand-crafted features, we adopted the same ones used by [Shardlow et al. \(2020\)](#) in the original evaluation of their dataset: **Logarithmic Frequency**, **Word Length** and **Syllable Length**. The latter two have been extracted using the Python `textstat` for each target word. As for the frequency feature, we extracted a general, out-of-domain frequency for each target word using the SUBTLEX database ([Brysbaert and New, 2009](#)) and the `wordfreq` Python package ([Speer et al., 2018](#)), and then we extracted the frequency of the word in each one of the three corpora composing CompLex. In total, we obtained 6 features (4 frequency + 2 length features) for each instance. We also added two Boolean features for **Capitalization**: the first was equal to 1 if the first letter of the target word was upper case and 0 otherwise; the second one was equal to 1 if all the letters of the target word were upper case and 0 otherwise. The latter feature was added because we noticed that some of the target words in the dataset are acronyms.

Apart from the lexical information, **Syntactic Features** were explored for both single words and

multiword expressions. The StanfordNLP tools (Manning et al., 2014) were first used to acquire both the part-of-speech (POS) tags and dependency trees. POS tags of target words were manipulated using one-hot encoding, for a total of 20 POS-based features. On the other hand, directed and path from the target word to the root were extracted as dependency features. We concatenated all dependency tags to the root, using one-hot encoding once again to encode every distinct path as a single feature. In total, we generated 267 dependency paths features with this mechanism.

Another feature was based on **Word Embedding similarity**: first, we computed the sum of the embeddings for all the words preceding the target, as a sort of general representation of the sentence context<sup>1</sup>, and then we measured the cosine similarity with the embedding of the target word. If the target was a multiword expression, we summed the embeddings of the words composing it. As word embeddings, we used the publicly available Fast-Text vectors, pre-trained on the Wikipedia corpus (Bojanowski et al., 2017).<sup>2</sup>

We added one feature based on the **BERT Transformer Model** (Devlin et al., 2019)<sup>3</sup> by masking the target word in the original sentence and taking the probability value provided in output by the Softmax. For multiword expressions, we sequentially masked the words composing the target and took the average value.

Similarly, we used the **GPT-2 Transformer Model** (Radford et al., 2019)<sup>4</sup> to obtain a probability score for the full sentence, computed as the product of the probabilities of the single tokens.

The total number of extracted features is 300. Finally, we decided to generate polynomial features from our set, in order to exploit potential interactions. We used the PolynomialFeatures functionality of the scikit-learn Python package to generate interaction features of order 2, so that the final number of features that was fed to the regressors was 45151.

<sup>1</sup>The use of vector sum as a compositional function has been used in Distributional Semantics since Mitchell and Lapata (2010).

<sup>2</sup><https://fasttext.cc/docs/en/pretrained-vectors.html>.

<sup>3</sup>We used the BERT-large-cased model, in the implementation of the Happy Transformer library: <https://github.com/EricFillion/happy-transformer>.

<sup>4</sup>We used the GPT2-xl model, in the implementation of the lm-scorer package: <https://pypi.org/project/lm-scorer/>.

## 4.2 Regressors

We tested several regression algorithms, using the implementations in the scikit-learn Python package. The adopted scikit-learn API and the main hyper-parameters are listed below:

- **RR Ridge**: Ridge Regression solves a regression model where the loss function is the linear least squares function and regularization is given by the l2-norm. `alpha=1.0, normalize=True`.
- **MLP MLPRegressor**: Multi-layer Perceptron regressor optimizes the squared-loss using LBFGS or stochastic gradient descent. `hidden_layer_size=5, activation=identity, solver=adam`.
- **PLSR PLSRegression**: PLS Regression implements the PLS2 blocks regression in case of one dimensional response. `components=5`.
- **BRR BayesianRidge**: a Bayesian Ridge model implements the optimization of the regularization parameters lambda and alpha. `alpha_1, alpha_2==1.0e-6, lambda_1, lambda_2=1.0e-6`.
- **LR LinearRegression**: Linear Regression is trained based on ordinary least squares function. `normalize=True`.
- **RF RandomForestRegressor**: a Random Forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. `min_samples_split=2, min_samples_leaf=1`.
- **GBR GradientBoostingRegressor**: Gradient Boosting builds an additive model in a forward stage-wise fashion which allows for the optimization of arbitrary differentiable loss functions. `learning_rate=0.1, min_samples_split=2, min_samples_leaf=1`.

## 4.3 Metrics

The performance of the participating systems was evaluated in terms of Pearson correlation ( $r$ ) between the outputs and the human mean ratings. In the Results section, we also report the scores

for Spearman correlation ( $\rho$ ), Mean Absolute Error ( $MAE$ ), Mean Squared Error ( $MSE$ ) and R-Square ( $R^2$ ).

## 5 Results

We evaluated our system for two subtasks based on given trial datasets. For each regressor, we tuned hyper-parameters according to each subtask. Performance evaluation has been carried out in two aspects: the assessment of the overall correlation with human ratings and the analysis of the contribution of the features.

### 5.1 Complexity Prediction

Evaluation metrics are reported ranking by Pearson correlation in Table 2 and 3 for single words and multiword expressions, respectively.

Regressor	$r$	$\rho$	$MAE$	$MSE$	$R^2$
RR	0.34	0.36	0.100	0.017	0.097
MLP	0.39	0.46	0.099	0.017	0.083
PLSR	0.46	0.53	0.093	0.015	0.206
BRR	0.47	0.51	0.094	0.015	0.181
LR	0.48	0.54	0.092	0.015	0.208
RF	0.49	0.65	0.078	0.010	0.472
GBR	0.75	0.72	0.070	0.008	0.561

Table 2: Performance on single words prediction (Sub-Task 1).

Regressor	$r$	$\rho$	$MAE$	$MSE$	$R^2$
RR	0.26	0.28	0.128	0.021	0.083
MLP	0.28	0.37	0.117	0.019	0.091
BRR	0.40	0.42	0.112	0.017	0.151
PLSR	0.40	0.42	0.109	0.017	0.178
LR	0.41	0.42	0.110	0.016	0.183
RF	0.44	0.51	0.105	0.015	0.424
GBR	0.66	0.66	0.090	0.013	0.427

Table 3: Performance of multiword expressions prediction (Sub-Task 2).

Features	$r$	$\rho$	$MAE$	$MSE$	$R^2$
Hand-crafted	0.73	0.69	0.072	0.009	0.527
+Synt.	0.73	0.69	0.073	0.009	0.528
+Embs.	0.73	0.69	0.073	0.009	0.530
+Trans.	0.74	0.71	0.071	0.009	0.545
+ALL	0.75	0.72	0.070	0.008	0.561

Table 4: Ablation study of feature groups.

It can be observed that predicting the complexity of single words is naturally less difficult than multiword expression. Concerning the regression algorithm, gradient boosting regression outperforms other investigated methods by a large gap, while PLS regression, Bayesian ridge regression, linear

regression and random forest regression perform very similarly. Though PLSR has a worse Pearson correlation than BRR, its  $R^2$  and Spearman correlation are slightly better. Further studies about regressors brought some unexpected results for our feature based approaches: based on the features we selected, Ridge Regression performs worse than linear regression, suggesting that some features are not suitable for applying L2-norm.

### 5.2 Feature Study

As our proposed method heavily relies on feature selection, the acquired features are investigated in four groups: *Hand-crafted* (including Logarithmic Frequency, Word and Syllable Length and Capitalization), *Syntactic* (including the POS- and the Dependency-based features), *Embedding* and *Transformer* features. We adopted the features of the *Hand-crafted* group as baseline, and present a comparison between the performance of systems using the other features as add-on components. The scores in Table 4 refer to the performance on the single words dataset, by using GBR as a regressor.

According to Table 4, syntactic, embedding and transformer based features can all contribute to improve the prediction results. As expected, the combination of all feature type groups can achieve the best predicting capability.

Comparing with the baseline of hand-crafted features, syntactic and embedding features have very marginal contribution. Yet, it should not be neglected supplementing only transformer based features cannot achieve the maximum performance gain. This indicates that the interaction of the individual features can bring latent useful information to model, further revealing the complexity values of the target words.

## 6 Conclusion

In this paper, we presented the PolyU CBS-Comp system for lexical complexity prediction, which took part in the SemEval shared task 1. Our method, based on a combination of lexical, syntactic, embeddings and Transformers features, achieved a 0.754 correlation on single words and 0.659 on multiword expressions, when using Gradient Boosting as a regression algorithm.

Traditional hand-crafted features, followed by Transformer-based ones, seem to give the strongest contribution to the classification performance, which is further improved by adding feature in-

teractions to the input for the regressor.

For future studies on lexical complexity, we plan to further exploit the text genre information, for example by adding domain-adapted language model features (Van Schijndel and Linzen, 2018) to the information available to our models.

## Acknowledgments

We acknowledge GRF grant (CERG PolyU 15211/14E, PolyU 152006/16E, PolyU YW4H), The Hong Kong Polytechnic University Postdoctoral Fellowships Scheme Projects.

## References

- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, and Judith A Blake. 2012. Concept Annotation in the CRAFT Corpus. *BMC Bioinformatics*, 13(1):1–20.
- Philippe Blache. 2011. Evaluating Language Complexity in Context: New Parameters for a Constraint-based Model. In *Proceedings of the International Workshop on Constraints and Language Processing*.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.
- Marc Brysbaert and Boris New. 2009. Moving Beyond Kučera and Francis: A Critical Evaluation of Current Word Frequency Norms and the Introduction of a New and Improved Word Frequency Measure for American English. *Behavior Research Methods*, 41(4):977–990.
- Emmanuele Chersoni, Philippe Blache, and Alessandro Lenci. 2016. Towards a Distributional Model of Semantic Complexity. In *Proceedings of the COLING Workshop on Computational Linguistics for Linguistic Complexity*.
- Emmanuele Chersoni, Alessandro Lenci, and Philippe Blache. 2017. Logical Metonymy in a Distributional Model of Sentence Comprehension. In *Proceedings of \*SEM*.
- Emmanuele Chersoni, Enrico Santus, Alessandro Lenci, Philippe Blache, and Chu-Ren Huang. 2021. Not All Arguments Are Processed Equally: A Distributional Model of Argument Complexity. *Language, Resources and Evaluation*, pages 1–28.
- Christos Christodouloupoulos and Mark Steedman. 2015. A Massively Parallel Corpus: The Bible in 100 Languages. *Language, Resources and Evaluation*, 49(2):375–395.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.
- Benedetta Iavarone, Dominique Brunato, and Felice Dell’Orletta. 2021. Sentence Complexity in Context. In *Proceedings of the NAACL Workshop on Cognitive Modeling and Computational Linguistics*.
- Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *MT Summit*, volume 5, pages 79–86. Citeseer.
- Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL: System Demo*.
- John H McWhorter. 2001. The World’s Simplest Grammars Are Creole Grammars. *Linguistic Typology*, 5(2-3):125–166.
- Jeff Mitchell and Mirella Lapata. 2010. Composition in Distributional Models of Semantics. *Cognitive Science*, 34(8):1388–1429.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 Task 11: Complex Word Identification. In *Proceedings of SemEval*.
- Mikael Parkvall. 2008. The Simplicity of Creoles in a Cross-linguistic Perspective. *Language Complexity: Typology, Contact, Change*, pages 265–285.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models Are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8):9.
- Gabriele Sarti, Dominique Brunato, and Felice Dell’Orletta. 2021. That Looks Hard: Characterizing Linguistic Complexity in Humans and Language Models. In *Proceedings of the NAACL Workshop on Cognitive Modeling and Computational Linguistics*.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex—A New Corpus for Lexical Complexity Prediction from Likert Scale Data. In *Proceedings of the LREC Workshop on Tools and Resources to Empower People with READING Difficulties*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of SemEval*.
- Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. [Luminosinsight/wordfreq: v2.2](#).
- Marten Van Schijndel and Tal Linzen. 2018. A Neural Model of Adaptation in Reading. In *Proceedings of EMNLP*.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in Data Annotation and System Performance. In *Proceedings of the IJCNLP Workshop on NLP Techniques for Educational Applications*.

# LAST at SemEval-2021 Task 1: Improving Multi-Word Complexity Prediction Using Bigram Association Measures

Yves Bestgen

Laboratoire d'analyse statistique des textes - LAST  
Institut de recherche en sciences psychologiques  
Université catholique de Louvain  
Place Cardinal Mercier, 10 1348 Louvain-la-Neuve, Belgium  
yves.bestgen@uclouvain.be

## Abstract

This paper describes the system developed by the Laboratoire d'analyse statistique des textes (LAST) for the Lexical Complexity Prediction shared task at SemEval-2021. The proposed system is made up of a LightGBM model fed with features obtained from many word frequency lists, published lexical norms and psychometric data. For tackling the specificity of the multi-word task, it uses bigram association measures. Despite that the only contextual feature used was sentence length, the system achieved an honorable performance in the multi-word task, but poorer in the single word task. The bigram association measures were found useful, but to a limited extent.

## 1 Introduction

For more than half a century, many studies have been carried out to collect norms about formal and semantic properties of words, such as frequency of use, spelling regularity, familiarity, age of acquisition, or emotional valence (Proctor and Vu, 1999). Some of these properties can be easily harvested through automatic counting procedures applied to corpora. Other properties, such as familiarity or emotional valence, are obtained by requiring participants, often more than ten, to rate the words on these dimensions. In psycholinguistics, these norms have been mainly used for selecting experimental materials (Wilson, 1988). In computational linguistics, they are used in opinion mining, in the evaluation of foreign language skills and in text simplification for instance (Pang and Lee, 2008; Kyle et al., 2018). Obtaining lexical norms that require human evaluations is extremely costly in time and resources, which greatly reduces their size. However, huge norms are essential in applications (Bestgen, 1994). This observation has led to the development of automatic techniques to extend such

norms (Bestgen, 2002; Kamps et al., 2004; Esuli and Sebastiani, 2006; Bestgen and Vincze, 2012).

The Lexical Complexity Prediction (LCP) shared task at SemEval-2021 requires exactly the development of such techniques (Shardlow et al., 2021a). It is indeed a question of estimating the lexical complexity, the degree of difficulty of the words in a text. This dimension is important in NLP applications for simplifying texts and assisting specific populations such as people with reading disabilities or who are learning a foreign language. A specificity of the LCP task is that it relates not only to words but also to multi-word expressions which are very rarely taken into account in norms and in automatic extension techniques (Bestgen, 2014). Another important feature of the task is that the target tokens were presented to human judges in context and that a significant number of them were presented in several different contexts. Human annotations are therefore likely to reflect the impact of the linguistic context on lexical complexity.

This paper describes the system proposed for this task by the Laboratoire d'analyse statistique des textes (LAST). It is based on a LightGBM model fed with features obtained from many word frequency lists, published lexical norms and psychometric data. For tackling the specificity of the multi-word task, it uses bigram association measures (such as Mutual Information) from research in lexicography (Church and Hanks, 1990) and in the automatic evaluation of texts written by English learners (Durrant and Schmitt, 2009; Bestgen and Granger, 2014). Despite that the only contextual feature used was sentence length, the system achieved an honorable performance in the multi-word task, ranking 9th out of 37 teams, but poorer in the single word task, ranking 26th out of 54 teams.

In the next section, the main characteristics of this challenge are summarized. The following sec-

tion describes in detail the developed system. Finally, the results in the challenge are reported along with several analyzes performed to get a better idea of the factors that affect the system performance.

## 2 Task and Materials

The organizers of the challenge have made available an updated version of the CompLex dataset (Shardlow et al., 2020) to the participating teams for developing their systems (i.e., the learning set). It consists of 8,083 single words and 1,616 bigrams, all of them presented in a one-sentence context. These sentences were taken from three English sources in almost equal proportion: biblical text, biomedical articles and proceedings of the European Parliament. The target words and bigrams were evaluated by several judges on a 5-point Likert scale depending on whether it seemed more or less easy to understand in this context. There were on average 25.75 annotations per instance (Shardlow et al., 2021b). The complexity score for each target is the mean of these ratings. In this materials, a non-negligible proportion of the targets were presented several times in different sentences in order to assess the impact of this context on the complexity assessment. The test set, collected in the same way, consisted of 917 single words and 184 bigrams of which none of the targets were present in the learning set. The challenge measure was Pearson’s linear correlation coefficient between human ratings and system predictions.

## 3 System

The first part of this section presents the features used to predict lexical complexity starting with those common to both tasks and ending with those specific to predicting the complexity of the multi-word expressions. Next, the procedure used to build the predictive models is described.

### 3.1 Features

**Frequency Lists:** I used the frequency of spelling forms calculated from corpora, but also a series of lists established by other researchers:

- The frequency in the Corpus of Contemporary American English (COCA), a balanced, 425-million word corpus of American English collected from 1990 to 2011 (<http://corpus.byu.edu/coca/>).

- The frequency in the British National Corpus (BNC), a 100-million word collection of samples of written and spoken language designed to represent a wide cross-section of British English from the latter part of the 20th century (<http://www.natcorp.ox.ac.uk/corpus/>).
- The Facebook frequency norms for American English and British English of Herdagdelen and Marelli (2017), based on approximately 1 billion tokens for each English variety, obtained from publicly available English posts collected between November 2014 and January 2015.
- The Rovereto Twitter Corpus frequency norms based on 75 millions tweets, for more than 1 billion tokens collected between December 2010 and July 2011 (Herdagdelen and Marelli, 2017).
- The USENET Orthographic Frequencies derived by Shaoul and Westbury (2006) from a corpus of 7,781,959,860 words of USENET postings collected between October 2005 and August 2006.
- The Hyperspace Analogue to Language (HAL) frequency norms provided by (Balota et al., 2007) for more that 40,000 words.
- The frequency word list derived from the Google’s ngram corpus available at <https://github.com/hackerb9/gwordlist>.

I also obtained the frequency of each target in each of the three corpora provided by the organizers as materials.

**Lexical Norms and Psychometric Data:** Lexical norms were mainly taken from the Glasgow Norms (Scott et al., 2019). They contain the evaluation by human raters of 5,553 English words on the psycholinguistic dimensions of age of acquisition, arousal, concreteness, dominance, familiarity, gender association, imageability, semantic size and valence. I also used SemD, a measure of the semantic ambiguity of a word based on variability in its contextual usage (Hoffman et al., 2013). The psychometric data were taken from the English Lexicon Project (Balota et al., 2007), a database that contains, for more than 40,000 words, the reaction time and average accuracy during lexical decision and naming tasks performed by many participants.



**Other Features:** Three binary features were used to encode the corpus from which the sentence is extracted, the initial analyzes having shown that it was more efficient than building three models, one per corpus. The only contextual feature taken into account was the sentence length in tokens.

**Bigram Association Measures:** These features, used only for the multi-word task, inform about the degree of association between the two target words according to a series of indices calculated on the basis of the frequency in a reference corpus of the bigram and that of the two words that compose it: pointwise mutual information and t-score (Church and Hanks, 1990), z-score (Berry-Rogghe, 1973), log-likelihood Chi-square test (Dunning, 1993), simple-ll (Evert, 2009), Dice coefficient (Kilgariff et al., 2014) and the two delta-p (Kyle et al., 2018). Bestgen and Granger (2014) refer to these features as *collgrams* because they combine the strengths of both collocations (by using association scores) and n-grams (by using contiguous pairs of words). The justification for their use in the LCP task is given by works in foreign language learning which has shown that these indices can be used to assess the lexical richness of multi-word expressions present in texts written by English learners (Bestgen and Granger, 2014; Somasundaran et al., 2015; Bestgen, 2018, 2019).

### 3.2 Supervised Learning Software

The regression models were built by the LightGBM open software (Ke et al., 2017), a well-known implementation of the gradient boosting decision tree approach. Compared to the multiple linear regression used for this task by Shardlow et al. (2020), this type of model has the advantages of not requiring any feature preprocessing, such as a logarithmic transformation, since it is insensitive to monotonic transformations. It also allows a very effective overfit control thanks to its many parameters.

### 3.3 Procedure

The sentences were first lemmatized by the Tree-Tagger (Schmid, 1994). The scores on the different lexical lists were attributed to the targets by a two-step procedure: on the basis of the orthographic form if it is found in the list or by using the lemma. The handling of missing values, which occurs when a word is not in a frequency list for example, has been left to the LightGBM default procedure. A large number of multi-word targets were given two

	Test	CV
	r	r
Full System	0.753	0.810
	Diff.	Diff.
Length	-0.002	-0.001
Frequencies	-0.013	-0.012
Normes	-0.022	-0.027

Table 1: Difference in Pearson’s r from the full system for the single word task when a set of features is removed (ablation approach).

values for many features by this procedure, one for each word. The corresponding features were doubled: the first encoding the minimum value and the second the maximum value.

The features used in the final models as well as LightGBM parameters were optimized by a 9-fold cross validation procedure. This led to the selection of the following features:

- For task 1, the length of the sentence and 12 features from the frequency lists, 10 from the lexical norms, and 8 from the psychometric data (i.e., average response latencies (raw and standardized), standard deviations, and accuracies for the lexical decision and naming tasks).
- For task 2, the same features as in task 1 plus 3 features for the corpus of origin and 8 from the bigram association measures.

The same LightGBM parameters were used for both tasks. They were left at their default values except the followings: *num\_iterations: 4800*, *max\_bin: 64*, *min\_data\_in\_bin: 10*, *lambda\_l2: 0.0175*, *bagging\_freq: 5*, *bagging\_fraction: 0.66*, *feature\_fraction: 0.09*, *learning\_rate: 0.0035*, *max\_depth: 7*, *min\_data\_in\_leaf: 7*, *num\_leaves: 11*.

## 4 Analyzes and Results

### 4.1 System Performance

The system built to predict the lexical complexity of single words scored 0.7534 on the test material, ranking it 26th out of 54 teams, down 0.0352 from the best team. In the multi-word subtask, the system finished 9th out of 37 teams with a score of 0.8417. The best team got 0.8612.

Line Id	Sent. Length	Corpus Id	Norms	Freq.	Bigram	Test	CV
						r	r
1	x	x	x	x	x	0.842	0.799
						Diff.	Diff.
2		x	x	x	x	-0.002	-0.001
3	x		x	x	x	-0.011	-0.003
4	x	x		x	x	-0.031	-0.015
5	x	x	x		x	-0.012	-0.004
6	x	x	x	x		-0.014	-0.014
7			x			-0.096	-0.055
8				x		-0.066	-0.054
9					x	-0.176	-0.161

Table 2: Difference in Pearson’s r from the full system for the multi-word task using the ablation approach.

The comparison of the results obtained on the test sets with those obtained by cross validation shows an unexpected difference between the two tasks. In the single word task, the correlation on the test set was lower by 0.053 compared to that obtained in CV (0.8064) while in the multi-word task this same correlation is higher by 0.042 compared to that obtained in CV (0.7996). It is also observed that the best systems which participated in the two tasks had superior performance on the multi-word task. If the difference in performance between the test sets and the CVs is not specific to the present system, this would suggest that the performance achieved in the multi-word task is rather overestimated, the test set being for some unknown reason relatively easy to predict. Although this is only a hypothesis which requires additional analyzes, it leads to not considering the multi-word task as being almost solved.

#### 4.2 Usefulness of the Different Types of Features

In this section, the impact of the different types of features on the system performance is assessed using an ablation procedure. As the previous section indicated important differences between performance on the test set and by the CV approach, results are presented for these two evaluation procedures.

**Single Word Task:** Table 1 shows that the sentence length, the only contextual feature, is of little use. Norms and psychometric data are more useful than frequencies in corpora, but above all, these two sets of features provide very similar informa-

tion since the removal of one as well as the other harms very little the model performance. These conclusions apply equally to the test set as to the CV.

**Multi-Word Task:** The system for multi-word expressions is based on five sets of features whose roles in its effectiveness are shown in Table 2. The absence of an "x" in a column indicates that this set of features has not been used in this version of the model. The first line of the table gives the performance of the system submitted for the challenge.

The length of the sentences [2] is much less useful than the features which identify the corpus [3]. The comparison of the usefulness of the psychometric norms and data and the frequencies in corpora shows a contrast. When these sets are in turn excluded from the system, psychometric norms and data [4] are more useful than frequencies in corpora [5]. On the other hand, when used alone, frequencies [8] are more effective than psychometric norms and data [7]. It will be concluded that a greater part of the contribution of the frequency data is shared with other indices, most probably the bigram association measures.

The specific contribution of the bigram association measures [6] to the performance of the system is slightly greater than that of the frequencies in corpora. These features provide a gain of 0.014. Without it, the system would have been ranked 15th instead of 9th in this task. When used alone, however, bigram association measures [9] are much less effective than norms or frequencies.

The effects of the different types of features are almost always more important when estimated on

Frequency of the target	Task	
	Single	Multi
1	49.2	87.4
2	19.9	7.8
3	10.2	2.2
4	6.2	0.8
5	13.5	1.8
6 or +	4.0	0.0
Total (%)	100.0	100.0
Total nbr.	3,850	1,479

Table 3: Percentage of the target frequency in the two tasks.

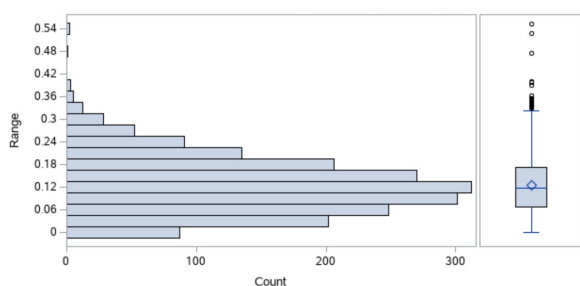


Figure 1: Distribution of the range for the repeated targets in the single-word task.

the test set rather than by CV. This could result from the initial difference in effectiveness between the two approaches. However, this phenomenon was not observed in the single-word task in which a difference in effectiveness was also observed. It is especially noted that the norms seem much more useful for the test set than for the CV.

**Potential Importance of the Context:** The results presented above indicate that, in both tasks, sentence length is of little use. Taking better account of the context is undoubtedly a way to improve the system. This hypothesis is all the more likely as the role of context could explain the difference in performance between the two tasks of this system, but also of those of the other teams. Two observations support this hypothesis. Firstly, an analysis of the target frequencies in the two tasks, presented in Table 3, shows that there are much more repeated targets in the single-word task than in the multi-word task, a statistically significant difference for a Chi-square test ( $p < 0.0001$ ).

Second, there are important differences between the human evaluations for the same target shown in different contexts. Figure 1 displays the distribu-

tion of the range (the difference between the maximum and the minimum values) of the complexity score for the repeated targets in the single-word task. The mean range is 0.125 and that 10% of the repeated targets have a range greater than 0.224. Being able to take these differences into account in the single-word task could significantly improve the system, provided that the differences in evaluation for the same target are not just noise. Only an analysis of the inter-rater reliability for the repeated targets would make it possible to choose between these two options.

## 5 Conclusion

The models proposed for the LCP task were built by the LightGBM software mainly fed with norms and frequency features. It obtained an acceptable performance on the test set in the multi-word task on the basis of little contextual information, but less so in the single word task. The analyzes carried out by a CV approach showed, on the other hand, that the system is no better in the multi-word task. It is therefore possible or even probable that the better performance results from an overestimation of its effectiveness. The bigram association measures (aka CollGrams) have proven to be useful, but to a limited extent.

Taking the context into account would probably have improved the system, especially for the single word task in which more than half of the targets were repeated. This hypothesis, however, is based on the assumption that differences between human ratings for the same target in different contexts are as reliable as their ratings for different targets. More generally, it would be interesting to explain the origin of the very important difference in performance between the two tasks, but that does not seem possible on the basis of the data I have access to.

## Acknowledgments

The author is a Research Associate of the Fonds de la Recherche Scientifique (FRS-FNRS).

## References

- David A. Balota, Melvin J. Yap, Keith A. Hutchison, Michael J. Cortese, Brett Kessler, Bjorn Loftis, James H. Neely, Douglas L. Nelson, Greg B. Simpson, and Rebecca Treiman. 2007. [The English lexicon project](#). *Behavior Research Methods*, 39:445–459.

- Godelieve L. M. Berry-Rogghe. 1973. The computation of collocations and their relevance in lexical studies. In Adam J Aitken, Richard W. Bailey, and Neil Hamilton-Smith, editors, *The Computer and Literary Studies*. Edinburgh University Press.
- Yves Bestgen. 1994. Can emotional valence in stories be determined from words? *Cognition and Emotion*, 7:21–36.
- Yves Bestgen. 2002. Détermination de la valence affective de termes dans de grands corpus de textes. In *Actes du Colloque International sur la Fouille de Texte CIFT'02*, pages 81–94, Nancy. INRIA.
- Yves Bestgen. 2014. Construction automatique d'un lexique de n-grammes pour la fouille d'opinion : Faisabilité et utilité. *Document Numérique*, 17:103–123.
- Yves Bestgen. 2018. Beyond single-word measures: L2 writing assessment, lexical richness and formulaic competence. *System*, 69:65–78.
- Yves Bestgen. 2019. Evaluation de textes en anglais langue étrangère et séries phraséologiques : comparaison de deux procédures automatiques librement accessibles. *Revue française de linguistique appliquée*, 24:81–94.
- Yves Bestgen and Sylviane Granger. 2014. Quantifying the development of phraseological competence in L2 English writing: An automated approach. *Journal of Second Language Writing*, 26:28–41.
- Yves Bestgen and Nadja Vincze. 2012. Checking and bootstrapping lexical norms by means of word similarity indexes. *Behavior Research Methods*, 44:998–1006.
- Kenneth Ward Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.
- Ted Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.
- Philip Durrant and Norbert Schmitt. 2009. To what extent do native and non-native writers make use of collocations? *International Review of Applied Linguistics in Language Teaching*, 47:157–177.
- Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).
- Stefan Evert. 2009. Corpora and collocations. In Anke Lüdeling and Merja Kytö, editors, *Corpus Linguistics. An International Handbook*, pages 1211–1248. Mouton de Gruyter.
- Amac Herdagdelen and Marco Marelli. 2017. Social media and language processing: How facebook and twitter provide the best frequency estimates for studying word recognition. *Cognitive Science*, 41:976–995.
- Paul Hoffman, Matthew A. Lambon Ralph, and Timothy T. Rogers. 2013. Semantic diversity: A measure of semantic ambiguity based on variability in the contextual usage of words. *Behavior Research Methods*, 45:998–1006.
- Jaap Kamps, Maarten Marx, Robert J. Mokken, and Maarten de Rijke. 2004. Using WordNet to measure semantic orientations of adjectives. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.
- Adam Kilgarriff, Vít Baisa, Jan Bušta, Miloš Jakubíček, Vojtěch Kovář, Jan Michelfeit, Pavel Rychlý, and Vít Suchomel. 2014. The Sketch Engine: ten years on. *Lexicography*, 1(1):7–36.
- Kristopher Kyle, Scott Crossley, and Cynthia Berger. 2018. The tool for the automatic analysis of lexical sophistication (TAALES): version 2.0. *Behavior Research Methods*, 50:1030–1046.
- Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1):1–135.
- Robert W. Proctor and Kim-Phuong L. Vu. 1999. Index of norms and ratings published in the psychological society journals. *Behavior Research Methods*, 31:659–667.
- Helmuth Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49.
- Graham G. Scott, Anne Keitel, Marc Becirspahic, Bo Yao, and Sara C. Sereno. 2019. The Glasgow norms: Ratings of 5,500 words on nine scales. *Behavior Research Methods*, 51:1258–1270.
- Cyrus Shaoul and Chris Westbury. 2006. USNET orthographic frequencies for 111,627 English words.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. Complex: A new corpus for lexical complexity prediction from likert scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. [Predicting lexical complexity in english texts](#).

Swapna Somasundaran, Chong Min Lee, Martin Chodorow, and Xinhao Wang. 2015. [Automated scoring of picture-based story narration](#). In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 42–48, Denver, Colorado. Association for Computational Linguistics.

Michael Wilson. 1988. [MRC psycholinguistic database: Machine-usable dictionary, version 2.00](#). *Behavior Research Methods*, 20:6–10.

# DeepBlueAI at SemEval-2021 Task 1: Lexical Complexity Prediction with A Deep Ensemble Approach

Chunguang Pan Bingyan Song Shengguang Wang Zhipeng Luo  
DeepBlue Technology (Shanghai) Co., Ltd  
{songby, panchg, wangshg, luozp}@deepblueai.com

## Abstract

Lexical complexity plays an important role in reading comprehension. lexical complexity prediction (LCP) can not only be used as a part of Lexical Simplification systems, but also as a stand-alone application to help people better reading. This paper presents the winning system we submitted to the LCP Shared Task of SemEval 2021 that capable of dealing with both two subtasks. We first perform fine-tuning on numbers of pre-trained language models (PLMs) with various hyperparameters and different training strategies such as pseudo-labelling and data augmentation. Then an effective stacking mechanism is applied on top of the fine-tuned PLMs to obtain the final prediction. Experimental results on the Complex dataset show the validity of our method and we rank first and second for subtask 2 and 1.

## 1 Introduction

Lexical complexity is one of the main reasons leading to overall text complexity and thus result in poor reading comprehension for readers (DuBay, 2004). Different from the Complex Word Identification (CWI) (Shardlow, 2014) task, which aims to predict whether a given word is complex or not, the goal of **lexical complexity prediction** (LCP) is to predict the complexity value of the given parts from contexts as shown in Figure 1. The underlined parts of the sentence are the words that need to be predicted and the same words in different contexts may have different complexity scores. LCP plays an important role in the usual Lexical Simplification (LS) (Bott et al., 2012) pipeline since it can help simplifiers find the challenging words and replace them with appropriate alternatives that easy to understand. Either LCP or CWI can not only be used as a component of LS systems but also as a stand-alone application within intelligent

Single word	Multi-words
<i>Context1:</i> They shall be to you for a <u>refuge</u> from the avenger of blood. <i>Complexity score:</i> 0.3475	<i>Context1:</i> SEM confirmed many of the observations made by <u>confocal microscopy</u> . <i>Complexity score:</i> 0.64473
<i>Context2:</i> There will be a pavilion for a shade in the daytime from the heat, and for a <u>refuge</u> and for a shelter from storm and from rain. <i>Complexity score:</i> 0.075	<i>Context2:</i> SJ and SVJ carried out <u>confocal microscopy</u> on whole-mounts of stria vascularis. <i>Complexity score:</i> 0.7750

Figure 1: Examples of LCP including single words and multi-words. The complexity score is the score for the underlined words.

tutoring systems for second language learners or in reading devices for people with low literacy skills (Gooding and Kochmar, 2018).

In this paper, we introduce our system for the lexical complexity prediction task of the SemEval-2021 (Matthew et al., 2021). We fulfill this task by leveraging multiple pre-trained language models (PLM) with different training strategies. There are two main steps for our system: (i) fine-tuning numbers of heterogeneous PLMs, including BERT (Devlin et al., 2019), ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019) and ERNIE (Zhang et al., 2019), with various hyperparameters and training strategies, obtaining diverse models; (ii) applying an effective stacking mechanism on top of these PLMs to predict the final complexity scores.

Our experiments, merging PLMs in total, indicate that our method successfully utilizes weaker PLMs as well as high-performing PLMs. As a result, our system ranks second and first for Subtask 1 and 2 of LCP 2021, SemEval-2021.

## 2 Related Work

### 2.1 Lexical Complexity Prediction

There has been some work for the creation and evaluation of automatically graded vocabulary lists

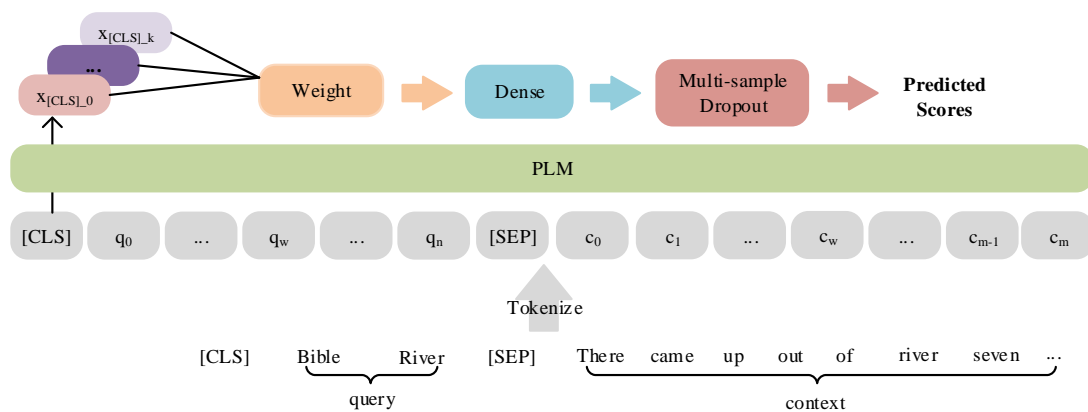


Figure 2: The overall architecture for predicting complexity scores.

for analyzing lexical complexity. François et al. (2014) present the first graded lexicon for French as a foreign language that reports word frequencies by difficulty level and Gala et al. (2014) train two SVM classifiers with 49 features, one for L1 learners and one for learners of French as a foreign language. Alfter and Volodina (2018) map the use of previously created word lists to a single CEFR scale (Common European Framework of Reference for Languages) (De l’Europe, 2003), then they add topics as additional features to predict the complexity level for learners of Swedish as a second language. Shardlow et al. (2020) point out the limitation of treating lexical complexity as a binary classification task. Therefore, they present the first English dataset for continuous lexical complexity prediction and develop a linear regression-based method with various features.

## 2.2 Complex Word Identification

A related area of LCP is CWI. Early studies on CWI either attempt to simplify all words (Thomas and Anderson, 2012) or set a frequency-based threshold (Biran et al., 2011). Shardlow (2013) indicates that a classification-based method to CWI is the most promising one. Most of the teams participating in two CWI shared tasks also use classification approaches with extensive feature engineering. In CWI 2016 (Paetzold and Specia, 2016a), complexity was defined as whether or not a word is difficult to understand for non-native English speakers and the words in the dataset are tagged as complex or non-complex by 400 non-native English speakers. The results highlight the effectiveness of Decision Trees (Quijada and Medero, 2016; Mukherjee et al., 2016) and Ensemble methods (Paetzold and Specia, 2016b; Malmasi et al., 2016) for the task.

In CWI 2018 (Yimam et al., 2018), a multilingual dataset was provided containing English, German, Spanish and French and there were two subtasks: binary classification and probabilistic classification. The submitted systems mainly use traditional machine learning classifiers (e.g. SVM, Random Forests) with features (Butnaru and Ionescu, 2018; Kajiwara and Komachi, 2018), deep learning methods (Hartmann and Dos Santos, 2018; De Hertog and Tack, 2018) and ensemble methods (Gooding and Kochmar, 2018; Aroyehun et al., 2018). More recently, (Gooding and Kochmar, 2019) propose a new perspective by treating CWI as a sequence labeling task that can detect both complex words and phrases. All these methods are different from ours which utilizes heterogeneous PLMs with various training strategies.

## 3 Background

**Task Definition** There are two subtasks in the LCP task. For subtask 1, the goal is to predict the complexity score for a single word from the given context. As an example shown in Figure 1, the ‘refuge’ is the word that needs to be predicted and since the meaning of it is harder to get in the first context, its complexity score in the first context is much higher. For subtask 2, the goal is to predict the complexity score for a multi-word expression from the given context. An example is also shown in the right part of Figure 1.

**Dataset** Shardlow et al. (2020) introduce a new English corpus, Complex, as the dataset for the LCP task of SemEval-2021. Instead of assigning binary scores for lexical complexity, they use crowdsourcing to annotate 8979 instances covering three genres with lexical complexity scores using

Parameters	BERT <sub>LARGE</sub>	ALBERT <sub>XXLARGE</sub>	RoBERT <sub>LARGE</sub>	ERNIE <sub>LARGE</sub>
batch_size	16	16	16	16
learning rate	5e-6	5e-6	5e-6	5e-6
hidden_layer	3	3	1, 3, 5	3, 5
dropout	0.2, 0.3	0.2, 0.3	0.2, 0.1; 0.2, 0.5; 0.2, 0.3	0.2, 0.3; 0.2, 0.5
loss function	MSE	MSE	RMSE, MSE, MAE	MSE, MAE

Table 1: Parameter settings for different base models

a 5-point Likert scale: one for very easy, two for easy, three for neutral, four for difficult, and five for very difficult. The numerical labels were transformed to a 0-1 range as shown in Figure 1. To add further variation to the data, three corpora were selected including Bible, Europarl (Koehn, 2005) and Biomedical (Bada et al., 2012). Each corpus has its own unique language features and styles. In addition to single words, multi-word expressions were also selected for annotating. In the end, there were 9476 annotated contexts with 5166 unique words.

## 4 System

### 4.1 PLMs-based Method

PLMs such as BERT (Bidirectional Encoder Representations from Transformers) use the encoder structure of the Transformer (Vaswani et al., 2017) for deep self-supervised learning, which requires task-specific fine-tuning. In this paper, the downstream task is to predict the complexity scores, a real-value in the range of [0,1], of given words. Our method is capable of dealing with both subtask 1 and 2. Figure 2 shows the main architecture of our BERT-based model for predicting complexity scores.

Since PLMs can process multiple input sentences, we add a query sentence before the context to emphasize the words (e.g. river) that need to be predicted and the corpus (e.g. Bible) they come from. We add special tokens [CLS] and [SEP] to separate the query and the context as shown in Figure 2. BERT first tokenizes the input contents and then generates contextualized vector representations for each token in multiple hidden layers. We focus on the output of only the first position that we passed the special [CLS] token to. The last  $k$  hidden layers are selected to get the final representation of token [CLS] through a weighted calculation function as below,

$$x_{[CLS]} = \sum_{i=1}^k W_i x_{[CLS]i}$$

where  $W_i$  is the learning weight for each hidden layer. The calculated representation is then fed into a dense layer, and the technique of multi-sample dropout (Inoue, 2019) is utilized to accelerate training and finally obtain the predicted complexity scores. The loss function can be chosen among several options including Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE).

### 4.2 Training strategies

In order to further improve the diversity of trained models, we incorporate two training strategies as depicted below.

**Pseudo-Labeling** Pseudo-labeling is the process of using a labeled data model to predict labels for unlabeled data. We predict the unlabeled test dataset and mix these pseudo labels with the training set together to train the new model.

**Data augmentation** Data augmentation is the technique used to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. It acts as a regularizer and helps reduce overfitting when training a machine learning model. In this paper, data augmentation consists of two parts. We first add the dataset released by CWI 2018 into the training set. Besides, for subtask 2, since its training dataset is small which only contains one thousand samples, we add the dataset of subtask 1 to train the model for subtask 2. Then, for a given sentence in the training set, we perform the operations containing synonym replacement, random insertion, random swap, and random deletion introduced by Wei and Zou (2019).

### 4.3 Stacking Trained Models

Model stacking is an efficient ensemble method to improve model accuracy. The main procedure of stacking trained models in our method including five steps. First, we use heterogeneous PLMs including BERT, RoBERTa, ALBERT, and ERNIE as base models. Second, we generate multiple



Scheme	Model	R	Rho	MAE	MSE	R2
Baseline	Complexity average	-	-	0.1049	0.0189	0.0007
	Log Frequency and Length	0.5376	0.5251	0.0867	0.0135	0.2864
BERT	ERNIE <sub>LARGE</sub>	0.7838	0.7321	0.0647	0.0069	0.6120
	ALBERT <sub>XXLARGE</sub>	0.7850	0.7332	0.0644	0.0069	0.6115
	BERT <sub>LARGE</sub>	0.7862	0.7296	0.0672	0.0073	0.5849
	RoBERTa <sub>LARGE+PL</sub>	0.7770	0.7279	0.0656	0.0070	0.6023
	RoBERTa <sub>LARGE+DA</sub>	0.7870	0.7432	0.0670	0.0078	0.5598
	<b>RoBERTa<sub>LARGE</sub></b>	<b>0.7903</b>	<b>0.7356</b>	<b>0.0648</b>	<b>0.0068</b>	<b>0.6170</b>

Table 2: Comparison of different pre-trained language models with training strategies of subtask 1

Scheme	Model	R	Rho	MAE	MSE	R2
Baseline	Complexity average	-	-	0.1164	0.0219	0.0000
	Log Frequency and Length	0.6249	0.6162	0.0900	0.0136	0.3807
BERT	RoBERTa <sub>LARGE</sub>	0.7900	0.8002	0.0753	0.0092	0.6178
	ALBERT <sub>XXLARGE+sub1</sub>	0.7901	0.7952	0.0755	0.00929	0.6157
	<b>RoBERTa<sub>LARGE+sub1</sub></b>	<b>0.8101</b>	<b>0.8236</b>	<b>0.0715</b>	<b>0.0085</b>	<b>0.6498</b>
Ensemble	mean	0.8252	0.8343	0.0690	0.0079	0.6739
	<b>LR</b>	<b>0.8330</b>	<b>0.8348</b>	<b>0.0678</b>	<b>0.0074</b>	<b>0.6892</b>

Table 3: Comparison of different pre-trained language models of subtask 2

hyperparameter sets by setting different values of dropout, selecting different numbers of last hidden layers, and using different loss functions. Since our purpose here is not only to find the best hyperparameter sets but also to collect diverse sets with reasonable performances, we keep all the training results from different sets. Third, we perform 7-fold cross-validation during the whole training process to avoid overfitting or selection bias. Fourth, we adopt several training strategies including using pseudo-labelling (Iscen et al., 2019) and data augmentation to further improve the diversity of trained models.

Ultimately, we train a simple linear regression model as the final estimator. Suppose that the complexity score predicted by a based model with one hyperparameter set is  $\hat{y}_j$ , then the final complexity scores will be calculated as below,

$$\hat{y} = \sum_{j=1}^N W_j \hat{y}_j$$

where  $N$  is the total number of various fine-tuned PLMs with different hyperparameters sets and  $W_j$  is the weight for each predicted score from different PLMs learned by a linear regression model.

## 5 Experiments

### 5.1 Evaluation Metrics

As mentioned in the official evaluation procedure of LCP 2021, several evaluation metrics are chosen including Pearson correlation (R), Spearman correlation (Rho), Mean absolute error (MAE), Mean

squared error (MSE), and R-squared (R2). The final results are ranked using Pearson correlation.

### 5.2 Parameter settings

All models are implemented based on the open-source transformers library of hugging face (Wolf et al., 2020), which provides thousands of pre-trained models that can be quickly downloaded and fine-tuned on specific tasks. Table 1 shows the four employed PLMs and different parameters we set for each PLM including different numbers of hidden layers, different dropout pairs, and different loss functions.

## 6 Results

### 6.1 Ablation Study

**PLMs with Training Strategies** For subtask 1, we use different PLMs including ERNIE<sub>LARGE</sub>, ALBERT<sub>XXLARGE</sub>, BERT<sub>LARGE</sub>, RoBERTa<sub>LARGE</sub> as shown in Table 2. The results are the average scores of 7-fold cross-validation on the training dataset. Since RoBERTa<sub>LARGE</sub> performs best on this task, we further incorporate the training strategies including pseudo-labelling (PL) and data augmentation (DA) with it. However, for the training dataset, we find that by adding the training strategies, the results decrease a little bit.

For subtask2, we use two types of PLMs which are RoBERTa<sub>LARGE</sub> and ALBERT<sub>XXLARGE</sub>. The results shown in Table 3 are also obtained by averaging the scores of 7-fold cross-validation on the training dataset. Since we have added the dataset of subtask 1 into subtask 2, we also show the results

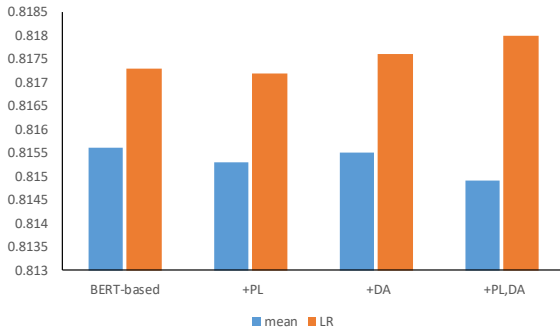


Figure 3: Comparison of Pearson Correlation values for stacking different models of subtask 1.

Subtask 1		Subtask 2	
System	R	System	R
qusaibanyismail	0.7886	<b>DeepBlueAI</b>	<b>0.8612</b>
<b>DeepBlueAI</b>	<b>0.7882</b>	rg_pa	0.8575
amsqr	0.7790	xiang_wen_tian	0.8571
armand.rotaru	0.7782	andi_gpu	0.8543
abdelkader	0.7779	ren_wo_xing	0.8541

Table 4: Leaderboard

of doing this in Table 3 and we can find that it is very effective by increasing 0.02 from base models.

**Stacking trained models** We use a linear regression (LR) model to stack different pre-trained models. We train the weights of each model in LR on the training set and then use the learning weights to predict the final scores of the test set.

Figure 3 shows the comparison of Pearson Correlation values for stacking different models of subtask 1. The columns in blue are the values computed by averaging predicted scores of different models while the columns in orange are the values through the LR function. We can clearly observe that the LR-based ensemble method outperforms those with the mean-based method, which verifies the validity of using the LR mechanism. Besides, although we find that adding training strategies to the base models would decrease performance according to Table 2, the performance will be improved when stacking them all. This indicates the positive effect of increasing model diversity.

## 6.2 Official Ranking

For both subtask 1 and subtask 2, among all the pre-submission experiments, we find that the scores obtained from stacking all the models performed best. The official ranking is presented in Table 4 and it demonstrates that our system is ranked first in subtask 2 and ranked second in subtask 1.

## 7 Conclusion

In this paper, we propose a top-performing model for the task of lexical complexity prediction. We fine-tune several pre-trained language models including BERT, ALBERT, RoBERTa, and ERNIE with different training strategies such as pseudo-labelling and data augmentation and stack them with a simple linear regression model. Experimental results show the effectiveness of this ensemble method and we win first place and second place for subtask 2 and 1.

## References

- David Alfter and Elena Volodina. 2018. Towards single word lexical complexity prediction. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 79–88.
- Segun Taofeek Aroyehun, Jason Angel, Daniel Alejandro Pérez Alvarez, and Alexander Gelbukh. 2018. Complex word identification: Convolutional neural network vs. feature engineering. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 322–327.
- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):1–20.
- Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501.
- Stefan Bott, Luz Rello, Biljana Drndarević, and Horacio Saggion. 2012. Can spanish be simpler? lexis: Lexical simplification for spanish. In *Proceedings of COLING 2012*, pages 357–374.
- Andrei Butnaru and Radu Tudor Ionescu. 2018. Unibuckernel: A kernel-based learning method for complex word identification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 175–183.
- Dirk De Hertog and Anaïs Tack. 2018. Deep learning architecture for complexword identification. In *Thirteenth Workshop of Innovative Use of NLP for Building Educational Applications*, pages 328–334. Association for Computational Linguistics (ACL); New Orleans, Louisiana.
- Conseil De l’Europe. 2003. *Cadre européen commun de référence pour les langues: apprendre, enseigner, évaluer*. Council of Europe.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- William H DuBay. 2004. The principles of readability. *Online Submission*.
- Thomas François, Núria Gala, Patrick Watrin, and Cédric Fairon. 2014. Flelex: a graded lexical resource for french foreign learners. In *International conference on Language Resources and Evaluation (LREC 2014)*.
- Nuria Gala, Thomas François, Delphine Bernhard, and Cédric Fairon. 2014. A model to predict lexical complexity and to grade words (un modèle pour prédire la complexité lexicale et graduer les mots)[in french]. In *Proceedings of TALN 2014 (Volume 1: Long Papers)*, pages 91–102.
- Sian Gooding and Ekaterina Kochmar. 2018. Camb at cwi shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- Sian Gooding and Ekaterina Kochmar. 2019. Complex word identification as a sequence labelling task. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153.
- Nathan Hartmann and Leandro Borges Dos Santos. 2018. Nilc at cwi 2018: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340.
- Hiroshi Inoue. 2019. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*.
- Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5070–5079.
- Tomoyuki Kajiwara and Mamoru Komachi. 2018. Complex word identification based on frequency in a learner corpus. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 195–199.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. Ltg at semeval-2016 task 11: Complex word identification with classifier ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000.
- Shardlow Matthew, Evans Richard, Paetzold Gustavo, and Zampieri Marcos. 2021. Semeval2021 task 1 : Lexical complexity prediction.
- Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Ju\_nlp at semeval-2016 task 11: Identifying complex words in a sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 986–990.
- Gustavo Paetzold and Lucia Specia. 2016a. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Gustavo Paetzold and Lucia Specia. 2016b. Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974.
- Maury Quijada and Julie Medero. 2016. Hmc at semeval-2016 task 11: Identifying complex words using depth-limited decision trees. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1034–1037.
- Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.
- Matthew Shardlow. 2014. Out in the open: Finding and categorising errors in the lexical simplification pipeline. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1583–1590.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. Complex: A new corpus for lexical complexity prediction from likert scale data. In *1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI) PROCEEDINGS Edited by Nuria Gala and Rodrigo Wilkens*, page 57.
- S Rebecca Thomas and Sven Anderson. 2012. Wordnet-based lexical simplification of a document. In *KONVENS*, pages 80–88.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications (NAACL 2018 Workshops)*, pages 66–78. Association for Computational Linguistics; Stroudsburg,.
- Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451.

# CS-UM6P at SemEval-2021 Task 1: A Deep Learning Model-based Pre-trained Transformer Encoder for Lexical Complexity

Nabil El Mamoun<sup>1</sup>   Abdelkader El Mahdaouy<sup>2</sup>   Abdellah El Mekki<sup>2</sup>  
Kabil Essefar<sup>2</sup>   Ismail Berrada<sup>2</sup>

<sup>1</sup>Faculty of Sciences Dhar EL Mahraz, Sidi Mohamed Ben Abdellah University, Morocco

<sup>2</sup>School of Computer Sciences, Mohammed VI Polytechnic University, Morocco

{firstname.lastname}@um6p.ma

## Abstract

Lexical Complexity Prediction (LCP) involves assigning a difficulty score to a particular word or expression, in a text intended for a target audience. In this paper, we introduce a new deep learning-based system for this challenging task. The proposed system consists of a deep learning model, based on a pre-trained transformer encoder, for word and Multi-Word Expression (MWE) complexity prediction. First, on top of the encoder's contextualized word embedding, our model employs an attention layer on the input context and the complex word or MWE. Then, the attention output is concatenated with the pooled output of the encoder and passed to a regression module. We investigate both single-task and joint training on both Sub-Tasks data using multiple pre-trained transformer-based encoders. The obtained results are very promising and show the effectiveness of fine-tuning pre-trained transformers for LCP tasks.

## 1 Introduction

Text Simplification (TS) is a fundamental task for improving text readability, and presents a wide variety of use cases, including assisting children with reading difficulties and native speakers with low literacy levels (De Belder and Moens, 2010; Aluísio and Gasperin, 2010), increasing accessibility for people with intellectual disabilities (Saggion, 2017), and facilitating certain aspects of language for language learners (Paetzold and Specia, 2016). TS may involve modifications to the syntactic structure of a sentence, its lexical units or both (Shardlow, 2014).

Lexical Simplification (LS), as a sub-task of TS, focuses on simplifying complex words of an input sentence. It first identifies complex words in a sentence, known as Complex Words Identification (CWI) or Lexical Complexity Prediction (LCP)

task. Then, it replaces them with other alternatives of equivalent meaning. Those substitutions should be more simplistic while preserving the semantic and the grammatical structure of the input sentence (Paetzold and Specia, 2017; Qiang et al., 2020).

Most of the previous research has modeled LCP as a binary classification task (Paetzold and Specia, 2017; Zampieri et al., 2016; Ronzano et al., 2016). A recent research study has introduced a multi-domain dataset, annotated using a 5-point Likert scale scheme (Shardlow et al., 2020). The aim is to label the complexity of a word or a Multi-Word Expression (MWE), in a more fine-grained manner, from very easy to very difficult. Hence, the lexical complexity of words is expressed on a continuous scale.

In this paper, we introduce our submitted system to the SemEval-2021 LCP 1 and 2 Sub-Tasks (Shardlow et al., 2021). The proposed system consists of a deep learning model for word and MWE complexity prediction. Our model employs a residual attention block and a regression module on top of a pre-trained transformer encoder, as follows:

- The encoder is fed with the concatenation of the context and the complex word or MWE, using the SEP token of the encoder's tokenizer.
- The residual attention block is a layer on top of the encoder's Contextualized Word Embedding (CWE) of the input context (sentence) and the complex word or MWE. The aim is to leverage the encoder's CWE to extract the relevant features of the inputs.
- The attention layer output is concatenated with the pooled output of the encoder and passed to the regression module for complexity prediction.

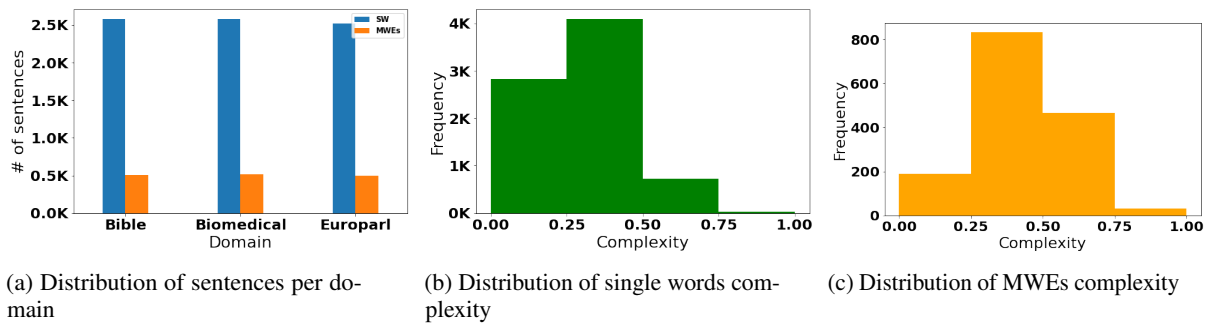


Figure 1: Domains and complexity distributions of the datasets sentences

The proposed model is trained to minimize both the Root Mean Square Error (RMSE) and the auxiliary loss associated to the negative Pearson Correlation. The two losses are combined using the uncertainty loss weighting (Kendall et al., 2017). We investigate two pre-trained transformer networks, namely BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). Moreover, we evaluate both single-task and joint training of word and MWE complexity prediction sub-tasks. The best performances are achieved using RoBERTa-large encoder while performing joint training on both Sub-Tasks data. The obtained results are very promising and show the effectiveness of our system, which was ranked among the top 10 submitted systems to both LCP 1 and 2 Sub-Tasks.

The rest of this paper is organized as follows. Section 2 describes the dataset and the sub-tasks of SemEval-2021 Task 1. In Section 3, we present our system overview. Section 4 summarizes and discusses the obtained results for both Sub-Task 1 and Sub-Task 2. Finally, Section 5 concludes the paper.

## 2 Task Description

### 2.1 Dataset Description

The dataset of the Lexical Complexity Prediction shared task (Shardlow et al., 2021) is an augmented version of the Complex dataset (Shardlow et al., 2020). In addition to complex word annotation, the data also include MWEs along with their context sentences and complexity scores. The dataset is annotated using a 5-point (1-5) Likert scale scheme and covers sentences from three domains: Bible, EuroParl, and Biomedical texts. The dataset is labeled by a group of annotators from English-speaking countries. It is compiled from sentences with at least four valid annotations. The aggregation of annotations is performed ensuring that

the normalized complexity is in the interval  $[0, 1]$ . The complexity scores are on a 5-point Likert scale and correspond to five levels of complexity ranging from "Very Easy" to "Very difficult" (Shardlow et al., 2020).

### 2.2 Sub-tasks Description

The LCP shared task consists of two sub-tasks (Shardlow et al., 2021):

- **Sub-Task 1:** predicting the complexity score of single words.
- **Sub-Task 2:** predicting the complexity score of multi-word expressions.

The training set consists of 7,662 samples for single word complexity prediction (Sub-Task 1), while the training set of MWE sub-task contains 1,517 samples (Sub-Task 2). Figure 1a presents the number of samples per domain. The dataset is almost balanced for all three domains in the two LCP sub-tasks. Figure 1b and 1c show the complexity distribution of single words and MWEs, respectively. The Figures (1b and 1c) illustrate that most single words have a complexity score less than 0.5, whereas for MWEs, the complexity scores are between 0.25 and 0.75.

## 3 System Overview

The proposed system uses a residual attention block and a regression module on top of the pre-trained transformer encoder network. In the following, we describe each component of our system.

### 3.1 Transformer Encoder

In order to encode the input context and the complex word or MWE, we employ two state-of-the-art pre-trained transformer encoder networks, namely BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019). First, the context (sentence) and the

complex word or MWE are concatenated using the special token (SEP or /s) of the encoder’s tokenizer, as follows:

- BERT case:

Input = [CLS] context [SEP] word/MWE

- RoBERTa case:

Input = < s > context < /s > word/MWE

Then, the tokenizer of the encoder splits the input into wordpieces  $[T_1, T_2, \dots, T_n]$  and encodes them using its vocabulary. The transformer encoder is fed with these encoded inputs. As a result, it outputs:

- The pooled embedding  $h_{pooled} \in \mathbb{R}^{1 \times d}$  (the embedding of [CLS] and < s > tokens for BERT and RoBERTa encoders, respectively).
- The contextualized word embedding (CWE)  $H = [h_1, h_2, \dots, h_n] \in \mathbb{R}^{n \times d}$  ( $d$  is the embedding dimension).

### 3.2 Attention block

Our model applies an attention layer on top of the CWE, output by the encoder (Bahdanau et al., 2015; Yang et al., 2016). The aim is to reward CWEs according to their relevance to the complexity prediction task. Using the CWE, the attention layer extracts a features vector  $v$ , representing the weighted sum of  $H$  vectors:

$$\begin{aligned} C &= \tanh(HW_a) \\ \alpha &= \text{softmax}(C^T W_\alpha) \\ v &= \alpha \cdot H^T \end{aligned}$$

where  $W_a \in \mathbb{R}^{d \times 1}$  and  $W_\alpha \in \mathbb{R}^{n \times n}$  are the learnable parameters of the attention layer,  $C \in \mathbb{R}^{n \times 1}$  is the context vector of the attention mechanism, and  $\alpha \in [0, 1]^n$  weights the CWEs according to their relevance to the task.

### 3.3 Regression Module

The regression module  $F$  consists of one hidden layer and one output layer.  $F$  is fed with the concatenation of the encoder’s pooled output  $h_{pooled}$  and the output attention block  $v$ .  $F$  outputs the  $\hat{y}$ , the predicted complexity:

$$\hat{y} = F([h_{pooled}, v])$$

The proposed system is trained to minimize both the Root Mean Square Error (RMSE) and the auxiliary loss associated to the negative Pearson Correlation:

- The RMSE loss:

$$\mathcal{L}_{rmse}(\hat{y}, y) = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

- The auxiliary loss associated to the negative Pearson Correlation:

$$\mathcal{L}_{aux}(\hat{y}, y) = 1 - \frac{\sum_{i=1}^N (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^N (y_i - \bar{y})^2} \sqrt{\sum_{i=1}^N (\hat{y}_i - \bar{\hat{y}})^2}}$$

where  $N$  is the number of samples,  $y$  is the ground truth complexity,  $\hat{y}$  is the predicted complexity, and  $\bar{y}$  (resp.  $\bar{\hat{y}}$ ) is the mean of  $y$  (resp.  $\hat{y}$ ). In order to combine both  $\mathcal{L}_{rmse}$  and  $\mathcal{L}_{aux}$ , we use the uncertainty loss weighting (Kendall et al., 2017). The latter aims to combine multiple losses according to their uncertainty and to avoid manual tuning of the loss weights. Finally, our model is trained to minimize the total loss, given by:

$$\mathcal{L}_{total} = \frac{1}{2\sigma_1^2} \mathcal{L}_{rmse} + \frac{1}{2\sigma_2^2} \mathcal{L}_{aux} + \log(\sigma_1 \sigma_2)$$

where  $\sigma_1$  and  $\sigma_2$  are two parameters for learning the relative weight of  $\mathcal{L}_{rmse}$  and  $\mathcal{L}_{aux}$ .

## 4 Results

This section describes the experiment settings and the obtained results.

### 4.1 Experiment Setting

We investigate the performance of our system using both the *base* and the *large* models of BERT and RoBERTa encoders:

- **BERT-base**: 12 transformer blocks,  $d = 768$ , 12 attention heads, and 110M parameters.
- **BERT-large**: 24 transformer blocks,  $d = 1024$ , 16 attention heads, and 336M parameters.
- **RoBERTa-base**: 12 transformer blocks,  $d = 768$ , 12 attention heads, and 125M parameters.
- **RoBERTa-large**: 24 transformer blocks,  $d = 1024$ , 16 attention heads, and 355M parameters.

We implement a simple text preprocessing pipeline that normalizes the contractions<sup>1</sup>. All models are trained using Adam optimizer (Kingma and

<sup>1</sup>We have employed the package contractions for this purpose. <https://github.com/kootenpv/contractions>

Training	Encoder	Sub-Task 1 (Word complexity)					Sub-Task 2 (MWE complexity)				
		Pearson	Spearman	MAE	MSE	R2	Pearson	Spearman	MAE	MSE	R2
Single Task	BERT-base	0.7211	0.7005	0.0742	0.009	0.4455	0.8199	0.8157	0.0732	0.0086	0.6703
	BERT-large	0.73	0.7023	0.0816	0.0105	0.3567	0.8214	0.8158	0.0723	0.0087	0.6668
	RoBERTa-base	0.7402	0.7198	0.0871	0.012	0.2654	0.8274	0.8235	0.0752	0.0087	0.6669
	RoBERTa-large	0.7613	0.7309	<b>0.0728</b>	<b>0.0088</b>	<b>0.4629</b>	0.8369	0.8349	0.0749	0.0088	0.6619
Joint Training	BERT-base	0.7236	0.7058	0.0827	0.0109	0.3288	0.8256	0.8125	0.0738	0.0088	0.6349
	BERT-large	0.7317	0.6936	0.077	0.0097	0.406	0.8371	0.8391	0.0703	0.0083	<b>0.7191</b>
	RoBERTa-base <sup>‡</sup>	0.7576	0.7318	0.0754	0.0091	0.4374	0.8424	0.8322	<b>0.0696</b>	<b>0.0078</b>	0.6767
	RoBERTa-large <sup>‡</sup>	<b>0.7779</b>	<b>0.7366</b>	0.0803	0.01	0.3813	<b>0.8489</b>	<b>0.8406</b>	0.076	0.0087	0.638

Table 1: The obtained results using single-task and joint training of both Sub-Tasks 1 and 2. The best performances are highlighted with bold font. The attached superscript ‡ denotes the results of our **two official submissions** to both Sub-Tasks 1 and 2 (TEST).

	Sub-Task 1 (Word complexity)					Sub-Task 2 (MWE complexity)				
	Pearson	Spearman	MAE	MSE	R2	Pearson	Spearman	MAE	MSE	R2
w/o attention	0.7584	0.7316	0.1089	0.0171	<b>0.485</b>	0.8323	0.8335	0.0941	0.094	<b>0.6632</b>
w/o auxiliary loss ( $\mathcal{L}_{aux}$ )	0.7597	0.7198	<b>0.0695</b>	<b>0.0082</b>	0.3176	0.8382	0.8352	<b>0.0692</b>	<b>0.0074</b>	0.6167
w/o uncertainty loss weighing	0.7694	0.7321	0.0728	0.0088	0.4623	0.8472	0.8401	0.0797	0.0103	0.6071
Model	<b>0.7779</b>	<b>0.7366</b>	0.0803	0.01	0.3813	<b>0.8489</b>	<b>0.8406</b>	0.076	0.0087	0.638

Table 2: Ablation study of our model’s component using joint training and RoBERTa-large as encoder (symbol w/o denotes without the corresponding component). **w/o uncertainty loss weighing** corresponds to the simple combination of model losses ( $\mathcal{L}_{total} = \mathcal{L}_{rmse} + \mathcal{L}_{aux}$ ).

Ba, 2015) with a learning rate of  $1 \times 10^{-5}$ . The batch size and the number of epochs are fixed to 16 and 5, respectively. We investigate both single-task training and joint training of both Sub-Task 1 and Sub-Task 2 (training a single model on both sub-tasks data). All models are trained on the full train sets, validated on the trial sets, and evaluated on the test set of each Sub-Task. For evaluation purpose, we use the shared task’s evaluation metrics, namely the **Pearson** correlation, the **Spearman** correlation, the Mean Absolute Error **MAE**, the Mean Squared Error **MSE**, and the coefficient of determination **R2**.

## 4.2 Experiment Results

Table 1 presents the obtained results of our model for both single-task and joint training, using the four transformer-based encoders. The overall results show that training joint models for both Sub-Tasks (1 and 2) outperform their single-task counterparts. The Use of deep encoders (large encoders) in our model yields better correlation performances. The best results for the correlation metrics are obtained using joint training and RoBERTa-large. For Sub-Task 1, the best MAE, MSE and R2 performances are achieved using single-task training and RoBERTa-large encoder. For Sub-Task 2, the best performances of all evaluation measures are obtained using joint training. In accordance with

Sub-Task 1, the best correlation performances are attained using RoBERTa-large encoder. Besides, the best R2 is achieved using BERT-large, while the top MAE and MSE performances are obtained using RoBERTa-base.

To sum up, the best performances are obtained by joint training of our model on top of a deep encoder. These results can be explained by the fact that deep encoders yield better input representation for both Sub-Tasks. The joint training helps to leverage signals from both Sub-Tasks.

## 4.3 Ablation Experiment

In order to assess the effectiveness of each component of our model, we perform an ablation study using joint training and RoBERTa-large encoder. Table 2 illustrates the results of our model’s ablation study. The results show that all components in our model improve the system performance. The auxiliary loss improves the performances of correlation measures, while it degrades MAE, MSE, and R2 performances. Combining RMSE and auxiliary losses using uncertainty loss weighting slightly improves the performance of correlation measures.

## 5 Conclusion

In this paper, we have presented our submitted system to the SemEval-2021 Task 1. The pro-



posed system consists of a deep learning model for word and MWE complexity prediction. Our model employs a residual attention block and a regression module on top of a pre-trained transformer encoder. We have trained the model to minimize the uncertainty weighted loss of the RMSE and the auxiliary loss associated to the negative Pearson correlation. Experiments are performed using the base and the large variants of the pre-trained BERT and RoBERTa encoders. The best performance is obtained using RoBERTa-large encoder while performing joint training on both Sub-Tasks data.

## References

- Sandra Aluísio and Caroline Gasperin. 2010. [Fostering digital inclusion and accessibility: The PorSimples project for simplification of Portuguese texts](#). In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53, Los Angeles, California. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Jan De Belder and Marie-Francine Moens. 2010. [Text simplification for children](#). In *Proceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM; New York.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. [Multi-task learning using uncertainty to weigh losses for scene geometry and semantics](#). *CoRR*, abs/1705.07115.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- G.H. Paetzold and L. Specia. 2017. [A survey on lexical simplification](#). *Journal of Artificial Intelligence Research*, 60:549–593.
- Gustavo H. Paetzold and Lucia Specia. 2016. [Unsupervised lexical simplification for non-native speakers](#). In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI’16*, page 3761–3767. AAAI Press.
- Jipeng Qiang, Y. Li, Y. Zhu, Yunhao Yuan, and X. Wu. 2020. [Lsbert: A simple framework for lexical simplification](#). *ArXiv*, abs/2006.14939.
- Francesco Ronzano, Ahmed AbuRa’ed, Luis Espinosa Anke, and Horacio Saggion. 2016. [TALN at semeval-2016 task 11: Modelling complex words by contextual, lexical and semantic features](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 1011–1016. The Association for Computer Linguistics.
- Horacio Saggion. 2017. [Automatic text simplification](#). *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.
- Matthew Shardlow. 2014. [A survey of automated text simplification](#). *International Journal of Advanced Computer Science and Applications(IJACSA), Special Issue on Natural Language Processing 2014*, 4(1).
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [Complex - A new corpus for lexical complexity prediction from likert scale data](#). *CoRR*, abs/2003.07008.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [Semeval-2021 task 1: Lexical complexity prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. [Hierarchical attention networks for document classification](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.
- Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. [Macsaar at semeval-2016 task 11: Zipfian and character features for complexword identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 1001–1005. The Association for Computer Linguistics.

# Cambridge at SemEval-2021 Task 1: An Ensemble of Feature-Based and Neural Models for Lexical Complexity Prediction

Zheng Yuan Gladys Tyen David Strohmaier

ALTA Institute, University of Cambridge, United Kingdom

Department of Computer Science and Technology, University of Cambridge, United Kingdom  
{firstname.lastname}@cl.cam.ac.uk

## Abstract

This paper describes our submission to the SemEval-2021 shared task on Lexical Complexity Prediction. We approached it as a regression problem and present an ensemble combining four systems, one feature-based and three neural with fine-tuning, frequency pre-training and multi-task learning, achieving Pearson scores of 0.8264 and 0.7556 on the trial and test sets respectively (sub-task 1). We further present our analysis of the results and discuss our findings.

## 1 Introduction

Predicting which words are considered hard to understand for a given target population has many applications. For example, it can be used to identify texts appropriate for language learners or included in a pipeline for automatic text simplification for people with low literacy skills or reading disabilities (Xia et al., 2016; Shardlow, 2014; Gooding and Kochmar, 2019b). In this paper, we describe our submission to the SemEval-2021 shared task on Lexical Complexity Prediction (LCP) (sub-task 1), where participating teams are expected to predict the complexity score of single words in context (Shardlow et al., 2021). Compared to previous shared tasks on Complex Word Identification (CWI), which have primarily focused on binary classification as systems were expected to identify words as complex or not (Paetzold and Specia, 2016a; Yimam et al., 2018); a new multi-domain English dataset was used for the purpose, which was annotated using a 5-point Likert scale (Shardlow et al., 2020). We approached LCP as a regression problem and proposed a traditional feature-based model, as well as three neural models exploring fine-tuning, frequency pre-training and multi-task learning (MTL).

The remainder of this paper is organised as follows. Section 2 presents related work in the area.

In Section 3, we describe our approach to the task and detail the four models included in our final ensemble system. In Section 4, we turn to the experiments, describing the data and evaluation metrics used, and presenting our results on the shared task trial set. Section 5 presents our official results on the shared task test set, and offers a discussion of the results and the performance of our submitted system. Finally, we conclude the paper and provide an overview of our findings in Section 6.

## 2 Related work

The SemEval-2016 shared task on CWI (Paetzold and Specia, 2016a) was framed as a binary classification problem, where complexity was defined as whether or not a word is difficult to understand for non-native English speakers. A set of 400 non-native speakers annotated the data in a binary fashion and a word was labelled as complex if it was annotated as complex by at least one annotator. The study performed by Zampieri et al. (2017) showed that most systems performed poorly due to the way the data was annotated. They also found out that words that were annotated as complex by the majority of human annotators tend to be easier for systems to identify, arguing that lexical complexity should be seen as a continuum on a spectrum rather than a binary value.

The second CWI shared task was organized as part of the BEA-2018 workshop (Yimam et al., 2018). It extended the previous one by introducing a new probabilistic classification sub-task where participants were asked to assign the probability that an annotator would find a word complex. The continuous complexity value for each word was calculated as the proportion of annotators that found a word complex. The results of the shared task showed that traditional feature engineering approaches (mostly based on length and

Linguistic feature	Pearson			Importance
	Train	Trial	Test	
1. Number of syllables in target word	<b>0.162</b>	<b>0.235</b>	<b>0.120</b>	0.019
2. Number of characters in target word	<b>0.099</b>	0.159	0.094	0.103
3. Mean number of syllables per word in given text	<b>0.057</b>	<b>0.224</b>	0.073	0.132
4. Mean number of characters per word in given text	<b>0.084</b>	<b>0.230</b>	0.089	0.119
5. Deviation from mean number of syllables	<b>0.079</b>	<b>-0.180</b>	-0.096	0.151
6. Deviation from mean number of characters	<b>-0.138</b>	-0.093	-0.062	0.180
7. Frequency of target word	<b>-0.240</b>	<b>-0.247</b>	<b>-0.183</b>	0.369
8. Frequency of target lemma	<b>-0.125</b>	<b>-0.256</b>	<b>-0.209</b>	0.714
9. Frequency of target dependency label	<b>-0.040</b>	-0.091	-0.054	0.059
10. Frequency of target POS	<b>0.051</b>	0.136	-0.010	0.128
11. Frequency of target word and dependency label	<b>-0.195</b>	-0.111	<b>-0.200</b>	0.089
12. Frequency of target word and POS	<b>-0.257</b>	<b>-0.201</b>	<b>-0.154</b>	0.126
13. Number of compounds in surrounding phrase	0.001	0.092	0.079	0.021
14. Number of modifiers in surrounding phrase	<b>0.073</b>	0.085	0.068	0.006
15. Number of dependencies linked to target word	<b>-0.073</b>	-0.044	-0.059	0.039

Table 1: Features used in the random forest regressor and the corresponding Pearson’s correlation with complexity in the training (train), trial, and test data; as well as the corresponding mean permutation importance ( $n = 50$ ). Bold Pearson values are significant ( $p < 0.001$ ).

frequency features) performed better than neural network and word embedding approaches, including the winning system **Camb-2018** from Gooding and Kochmar (2018). However, this system was subsequently outperformed by a sequence labeller approach to CWI that incorporated word context (Gooding and Kochmar, 2019a). In both shared tasks, the top-performing systems demonstrated the strength of ensemble models (Paetzold and Specia, 2016b; Gooding and Kochmar, 2018).

### 3 Approach

#### 3.1 Random forest regression

As a baseline, we began with training a simple random forest regressor based on 15 manually selected linguistic features. The regressor was trained with 100 trees, and we used mean absolute error (MAE) to measure the quality of each split. Most of our features were inspired by psycholinguistic studies and readability metrics. The full list of features can be found in Table 1.

**Frequency** Based on the psycholinguistic findings that the frequency of a word is strongly correlated with the speed at which it is processed (Preston, 1935; Monsell et al., 1989; Brysbaert et al., 2011), we introduced six features which are based on frequencies found in the Simple English

Wikipedia (SimpleWiki).<sup>1</sup> We selected SimpleWiki for its standardised form, relatively low frequency of complex words, and coverage of a large number of topics. Two of our frequency-based features were calculated based on the frequency of words that match both the surface form and the syntactic role - this was done as a coarse form of word sense disambiguation, but also to capture syntactic complexity.

**Syntax** Psycholinguistic studies have shown that syntactic complexity is linked to processing speed (Ferreira, 1991) and working memory limitations (Norman et al., 1992), which may affect participants’ perception of lexical complexity. In a similar vein, we added three syntactic features: the number of compounds and modifiers in the phrase containing the target word, and the number of child dependencies linked to the target word.

**Readability** We included syllable-based<sup>2</sup> and character-based metrics, which were inspired by traditional readability metrics such as the Flesch-Kincaid readability tests (Kincaid et al., 1975)

<sup>1</sup>The Simple English Wikipedia data can be accessed at <https://simple.wikipedia.org/>. For this shared task, we used a preprocessed version at <https://github.com/LGDoor/Dump-of-Simple-English-Wiki>.

<sup>2</sup>The number of syllables were estimated using the Syllables library: <https://github.com/prosegrinder/python-syllables>.

and the Coleman-Liau index (Coleman and Liau, 1975).

### 3.2 Fine-tuning BERT

Fine-tuning pre-trained language models via supervised learning has become the key to achieving state-of-the-art performance in various natural language processing (NLP) tasks. Our approach builds upon this, where we used BERT (Devlin et al., 2019) as the underlying language model and added a linear layer on top that allows for regression.

We treated it as a *sequence regression* problem and constructed the input by concatenating the target word  $w_t$ , the complexity of which was to be determined, and its context sentence:

$$[CLS]; w_t; [SEP]; w_1, \dots, w_t, \dots; [SEP] \quad (1)$$

We then fed the  $[CLS]$  representation into the output layer for regression.

We used the L1-loss, which measures the MAE for the prediction, i.e.:

$$Loss = mean(\{l_1, \dots, l_N\}); l_n = |x_n - y_n| \quad (2)$$

where  $x$  and  $y$  are respectively the output of the model and the target value.  $N$  is the batch size.

During training, the whole model was optimised in an end-to-end manner.

### 3.3 Frequency pre-training

We proposed an extension to the fine-tuning BERT system by introducing a pre-training step. We constructed a new pre-training set with 20,000 sentences extracted from SimpleWiki, filtering for whole sentences by detecting the presence of verbs, and removing sentences that are longer than 256 words, as this is the length of the longest sentence in the training data.

Frequency of each word and part-of-speech (POS) combination in SimpleWiki was counted and converted into a value between 0 and 1:

$$1 - \frac{\ln(f)}{\ln(h)} \quad (3)$$

where  $f$  is the original frequency value and  $h$  is the highest frequency found (excluding stop words). This conversion makes use of the Zipfian distribution observed in natural language (Zipf, 1935), allowing the model to be pre-trained on output values that match the range in the shared task dataset (see Section 4.1 for more details).

Data	Train	Trial	Test
Bible	2,574	143	283
Biomedical	2,576	135	289
Europarl	2,512	143	345
Total	7,662	421	917

Table 2: Number of single word instances in the training (train), trial and test subsets of the Bible, Biomedical and Europarl datasets.

We chose this particular frequency feature because it is the most strongly correlated one with the complexity values in the training data among the 15 features used in the random forest regressor (see Table 1 #12).

### 3.4 Neural multi-task learning

MTL allows models to use information from related tasks and learn from multiple objectives, which leads to performance improvement on individual tasks (Rei and Yannakoudakis, 2017; Yuan et al., 2019; Taslimipoor et al., 2020; Andersen et al., 2021). Instead of only predicting the complexity value of word in context, we extended the model to incorporate auxiliary objectives. We used a joint learning approach trained on in-domain data only and experimented with three related tasks to boost model performance:

- POS tagging
- Grammatical Relations (GR) prediction: We included as an auxiliary objective the prediction of the GR type of a dependent with its head.
- Genre classification: A classification task was introduced to predict the genre of the text.

Model weights were shared between the main and auxiliary training objectives. We used pre-trained DistilBERT (Sanh et al., 2020) for language representation as the basis for our neural network and added additional layers on top of the Transformer (Vaswani et al., 2017) architecture for fine-tuning.

The final layer for the LCP objective is a fully connected layer that performs regression. Different from the first two neural systems, we treated it as a *token regression* problem, where we only input the context sentence, and fed the vector representation of the target word  $w_t$  into the output layer for

Hyper-parameter	BERT	BERT <sub>freq.</sub>	MTL
Language model	bert-base-uncased	bert-base-uncased	distilbert-base-uncased
Max. length	190	160	304
Batch Size	40	8	1
Epochs	5	7	4
Decay rate	0.01	0.01	0.01
Learning rate	5e-06	2e-05	1e-05
Schedule	linear	linear	linear
Warm up steps	80	90	7662

Table 3: Hyper-parameters used for experiments.

regression:

$$[CLS]; w_1, \dots, w_t, \dots, w_N; [SEP] \quad (4)$$

For those cases where the target word was split into multiple sub-tokens, we took the averaged representation.

Additionally, a new output layer was introduced to perform the auxiliary task. For the first two token-level auxiliary tasks (POS and GR), the token representations were fed into the output layer. The model only predicted labels for auxiliary objectives on the first token of a word, in an identical fashion to Devlin et al. (2019). For genre classification, we used the  $[CLS]$  representation. The overall loss function is a weighted sum of the main LCP loss (measured as MAE) and the auxiliary loss (as cross-entropy):

$$Loss = \lambda Loss_{LCP} + (1 - \lambda) Loss_{aux} \quad (5)$$

## 4 Experiments

### 4.1 Dataset and evaluation

The data used in this shared task is an augmented version of CompLex (Shardlow et al., 2020), a multi-domain English dataset with sentences annotated using a 5-point Likert scale with 1 being very easy and 5 being very difficult. The final complexity labels were normalised in the range of  $[0, 1]$ . The dataset contains texts of three genres (Bible, Biomedical and Europarl) and both single words (sub-task 1) and multi-word expressions (sub-task 2). Since we focused on sub-task 1, we used only single word instances in our experiments. Corpus statistics are given in Table 2.

Systems were evaluated using Pearson correlation. We also report scores for the following metrics: Spearman correlation, MAE, mean squared error (MSE) and R-squared (R2).

### 4.2 Training details

We used spaCy<sup>3</sup> to preprocess the data and automatically generated lemma, POS and GR labels to be used in our experiments.

For the feature-based system, we used the random forest regressor in the *scikit-learn* library.<sup>4</sup> For the neural systems, we used pre-trained language models provided by *huggingface* (Wolf et al., 2020).<sup>5</sup> All neural systems were trained using the AdamW variant (Loshchilov and Hutter, 2019) of the Adam stochastic optimisation algorithm (Kingma and Ba, 2015). Detailed hyper-parameters are listed in Table 3. Each neural model was trained on one NVIDIA Tesla P100 GPU.

### 4.3 Individual system performance

Individual system performance on the trial set is reported in Table 4, where **RandomForest** refers to the feature-based random forest regression system, **BERT** refers to the fine-tuned BERT system, **BERT<sub>freq.</sub>** refers to the fine-tuned BERT system with frequency pre-training, and **MTL<sub>x</sub>** refers to the MTL system with the subscript ‘x’ representing the auxiliary task (POS, GR, or genre). We also report results from the winning system **Camb-2018** in the BEA-2018 CWI shared task, a feature-based, context-independent linear regression model.

We can see that our feature-based **RandomForest** system achieved comparable performance to the heavily feature-engineered **Camb-2018** system, despite using only 15 features. This may be due to the fact that linguistic features are often highly interdependent and capture very similar information.

We also notice that all our neural systems outperformed both feature-based systems by large margins (+0.1 Pearson). This contradicts the findings

<sup>3</sup><https://spacy.io/>

<sup>4</sup><https://scikit-learn.org/>

<sup>5</sup><https://huggingface.co/transformers/>

System	Pearson	Spearman	MAE	MSE	R2
RandomForest	0.7043	0.6746	0.0751	0.0096	0.4934
BERT	0.7907	<b>0.7579</b>	0.0647	0.0072	0.6191
BERT <sub>freq.</sub>	<b>0.8089</b>	0.7546	<b>0.0646</b>	<b>0.0068</b>	<b>0.6397</b>
MTL <sub>POS</sub>	0.8000	0.7528	0.0662	0.0075	0.6052
MTL <sub>GR</sub>	0.7936	0.7208	0.0654	0.0070	0.6290
MTL <sub>genre</sub>	0.7982	0.7272	0.0656	0.0070	0.6300
Camb-2018	0.7079	0.6885	0.0746	0.0095	0.4957

Table 4: Performance of individual systems on the trial set (sub-task 1). The best results are marked in bold. **Camb-2018** is the winning system in the BEA-2018 CWI shared task.

Ensemble	Pearson	Spearman	MAE	MSE	R2
MTL <sub>All</sub>	0.8129	0.7471	0.0634	0.0065	0.6542
MTL <sub>All</sub> + BERT + BERT <sub>freq.</sub>	0.8228	0.7641	<b>0.0621</b>	<b>0.0063</b>	0.6684
MTL <sub>All</sub> + BERT + BERT <sub>freq.</sub> + RandomForest	<b>0.8264</b>	<b>0.7676</b>	0.0623	<b>0.0063</b>	<b>0.6688</b>

Table 5: Performance of ensemble systems on the trial set (sub-task 1). The best results are marked in bold.

from the BEA-2018 CWI shared task where traditional feature-based approaches performed better than neural network and word embedding approaches. This could possibly be explained by the use of pre-trained Transformer-based language models in our neural systems, as well as a different annotation scheme employed when constructing the CompLex dataset used for this shared task. Nevertheless, our findings appear to match the general trend in NLP where neural systems are overtaking feature-based models as the state of the art. All our neural systems produced comparable results: **BERT<sub>freq.</sub>** yielded the best Pearson, MAE, MSE and R2 scores; while **BERT** yielded the best Spearman score.

#### 4.4 Ensemble performance

We further averaged the outputs from individual systems to obtain an ensemble. Table 5 shows results for different system combinations. Overall, the best system consists of all our individual systems proposed in Section 3, including the feature-based **RandomForest** system; and achieved the best Pearson score of 0.8264, Spearman of 0.7676, MSE of 0.0063, and R2 of 0.6688. The ensemble of all neural systems yielded the best MAE of 0.0621.

## 5 Official results and discussion

Our submission to the LCP shared task (sub-task 1) is the result of our best system (in terms of Pearson), an ensemble of three neural and one feature-based systems **MTL<sub>All</sub> + BERT + BERT<sub>freq.</sub> +**

**RandomForest**. The official results are reported in Table 6. Our final system achieved a Pearson score of 0.7556.

### 5.1 Per-genre performance

Using the Pearson correlation metric, the highest performance is obtained on the Biomedical data, followed by the Bible and Europarl data. On the MAE metric, however, the worst performance is found for the Biomedical data (see Table 6). We hypothesise that this might result from differences in the distribution of the lexical complexity scores. In particular, the scores for the Biomedical data appear to have a slightly larger interquartile range (see Appendix A, Figure A.1c).

### 5.2 Individual system contribution

To measure the contribution of each individual system to the overall performance, a number of ablation tests were performed, where one system was removed at a time. Results in Table 6 suggest that all neural systems have positive effects on the overall performance. Among them, **MTL<sub>All</sub>** is the most effective one, whose absence is responsible for a 0.02 decrease in Pearson, followed by **BERT<sub>freq.</sub>** and **BERT**. Interestingly, removing **RandomForest** yielded a better Pearson score of 0.7560, indicating that it is detrimental and brought performance down. This is inconsistent with our results on the trial set (see Table 5), where all systems contributed to the final system.

		Pearson	Spearman	MAE	MSE	R2
Official		0.7556	0.7105	0.0646	0.0070	0.5705
<b>Genre</b>	Bible	0.7475	0.7154	0.0662	0.0076	0.5493
	Biomedical	0.7763	0.7274	0.0745	0.0088	0.6025
	Europarl	0.7195	0.6699	0.0551	0.0049	0.5169
<b>Ablated system</b>	RandomForest	0.7560	0.7126	0.0647	0.0070	0.5685
	BERT	0.7523	0.7069	0.0650	0.0070	0.5655
	BERT <sub>freq.</sub>	0.7515	0.7067	0.0652	0.0071	0.5633
	MTL <sub>All</sub>	0.7371	0.6920	0.0669	0.0076	0.5335

Table 6: Official results of our submitted system on the test set (sub-task 1). Per-genre performance and ablation test results are included.

**Analysis of RandomForest** To understand why the feature-based regressor performed worse on the test data, we examined the correlation between each feature and the complexity scores in the training (train), trial, and test sets. Results in Table 1 show that several linguistic features (particularly #3, #4, and #10) are more strongly correlated with scores in the trial data compared to the test data, which may explain the discrepancy in our results. Although most features appear to have a small but significant correlation with complexity in the training data, many are not significant in the test data, likely due to the smaller sample size. This suggests that, while there may be some weak, overall correlation between these features and complexity, there is sufficient noise in the data that the relationship is negligible and unreliable when used to predict the complexity of a given word.

Additionally, we investigated the importance of each feature in the random forest regressor, as measured by the mean permutation importance (Breiman, 2001) - see Table 1. Our analysis reveals that the frequency of the target lemma (#8) is the most important one, followed by the frequency of the target word itself (#7). Both of these features are more strongly correlated with complexity in the trial data than either the training or test data, which also contributes to the inconsistency described above.

## 6 Conclusion

This paper presents our contribution to the SemEval-2021 shared task on LCP. We competed in sub-task 1 (single words) with an ensemble system combining three neural models and one feature-based model. Our analysis reveals that even though all three neural systems perform comparably, the MTL system contributed the most to the ensemble

system. Adding the feature-based model improved the performance on the trial data, but brought performance down on the test data. In addition to the mismatch between the trial and test data, the noise in the data further contributed to this inconsistency. The comparatively lower performance of the feature-based system is especially interesting in light that such systems were competitive for CWI until relatively recently (Gooding and Kochmar, 2018). When looking at different genres, our submitted system yielded the highest performance in Pearson, but worst performance in MAE in Biomedical domain, compared to the other genres. We hypothesise that this is due to differences in data distribution between genres.

## Acknowledgments

We thank Sian Gooding and Ekaterina Kochmar for support and advice. This paper reports on research supported by Cambridge Assessment, University of Cambridge. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery operated by the University of Cambridge Research Computing Service, provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council. We acknowledge NVIDIA for an Academic Hardware Grant.

## References

Øistein E. Andersen, Rebecca Watson, Zheng Yuan, and Kevin Yet Fong Cheung. 2021. Benefits of alternative evaluation methods for automated essay scoring. In *Proceedings of the 14th International Conference on Educational Data Mining (EDM 2021)*. International Educational Data Mining Society.

- Leo Breiman. 2001. [Random forests](#). *Machine Learning*, 45(1):5–32.
- Marc Brysbaert, Matthias Buchmeier, Markus Conrad, Arthur M. Jacobs, Jens Bölte, and Andrea Böhl. 2011. [The word frequency effect](#). *Experimental Psychology*, 58(5):412–424.
- Meri Coleman and Ta Lin Liau. 1975. [A computer readability formula designed for machine scoring](#). *Journal of Applied Psychology*, 60(2):283–284.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Fernanda Ferreira. 1991. [Effects of length and syntactic complexity on initiation times for prepared utterances](#). *Journal of Memory and Language*, 30(2):210–233.
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2019a. [Complex word identification as a sequence labelling task](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153, Florence, Italy. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2019b. [Recursive context-aware lexical simplification](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4853–4863, Hong Kong, China. Association for Computational Linguistics.
- J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. [Derivation of new readability formulas \(automated readability index, fog count and flesch reading ease formula\) for navy enlisted personnel](#). Technical report, Defense Technical Information Center.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *Proceedings of the International Conference on Learning Representations (ICLR 2015)*.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *Proceedings of the International Conference on Learning Representations (ICLR 2019)*.
- Stephen Monsell, Michael C Doyle, and Patrick N Haggard. 1989. [Effects of frequency on visual word recognition tasks: Where are they?](#) *Journal of Experimental Psychology: General*, 118(1):43–71.
- Suzanne Norman, Susan Kemper, and Donna Kynette. 1992. [Adults' reading comprehension: Effects of syntactic complexity and working memory](#). *Journal of Gerontology*, 47(4):P258–P265.
- Gustavo Paetzold and Lucia Specia. 2016a. [SemEval 2016 task 11: Complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016b. [SV000gg at SemEval-2016 task 11: Heavy gauge complex word identification with system voting](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974, San Diego, California. Association for Computational Linguistics.
- Katherine A. Preston. 1935. [The speed of word perception and its relation to reading ability](#). *The Journal of General Psychology*, 13(1):199–203.
- Marek Rei and Helen Yannakoudakis. 2017. [Auxiliary objectives for neural error detection models](#). In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 33–43, Copenhagen, Denmark. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter](#). In *Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing*, Vancouver BC, Canada.
- Matthew Shardlow. 2014. [Out in the open: Finding and categorising errors in the lexical simplification pipeline](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1583–1590, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [SemEval-2021 Task 1: Lexical Complexity Prediction](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.



Shiva Taslimipour, Sara Bahaadini, and Ekaterina Kochmar. 2020. [MTLB-STRUCT @parseme 2020: Capturing unseen multiword expressions using multi-task learning and pre-trained masked language models](#). In *Proceedings of the Joint Workshop on Multiword Expressions and Electronic Lexicons*, pages 142–148, online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is All you Need](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. [Text readability assessment for second language learners](#). In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, San Diego, CA. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaís Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. [Neural and FST-based approaches to grammatical error correction](#). In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 228–239, Florence, Italy. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. [Complex word identification: Challenges in data annotation and system performance](#). In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63, Taipei, Taiwan. Asian Federation of Natural Language Processing.

George Kingsley Zipf. 1935. [The Psycho-Biology Of Language](#). Routledge.

## A Data distribution

Figure A.1 presents the box plots of complexity scores in the training (train), trial and test subsets of the Bible, Biomedical and Europarl datasets.

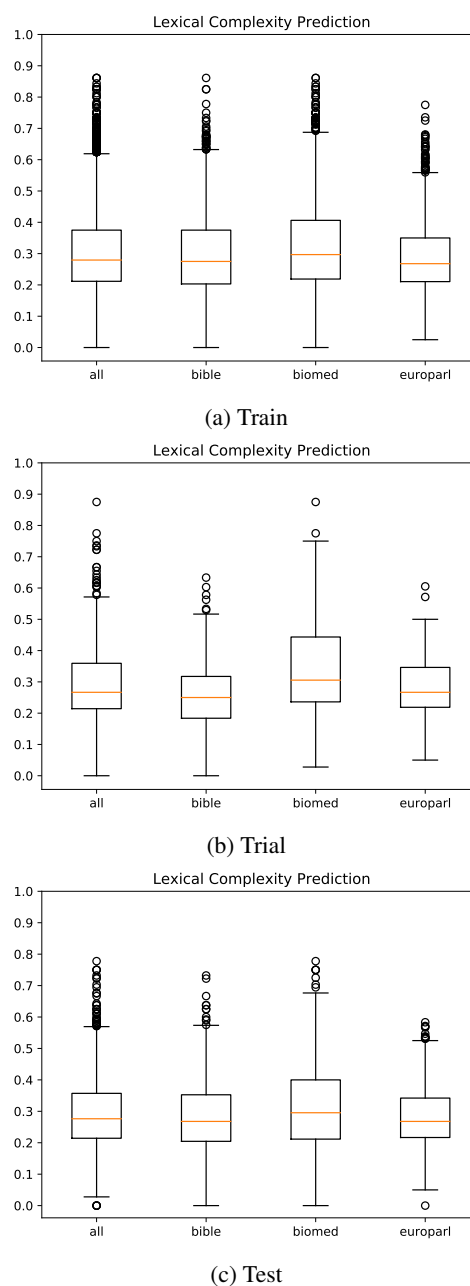


Figure A.1: Box plots of complexity scores for data by genre.

# hub at SemEval-2021 Task 1: Fusion of Sentence and Word Frequency to Predict Lexical Complexity

Bo Huang, Yang Bai, Xiaobing Zhou\*

School of Information Science and Engineering

Yunnan University, Yunnan, P.R. China

\*Corresponding author: zhouxb@ynu.edu.com

## Abstract

In this paper, we propose a method of fusing sentence information and word frequency information for the SemEval 2021 Task 1-Lexical Complexity Prediction (LCP) shared task. In our system, the sentence information comes from the RoBERTa model, and the word frequency information comes from the Tf-Idf algorithm. Use Inception block as a shared layer to learn sentence and word frequency information. We described the implementation of our best system and discussed our methods and experiments in the task. The shared task is divided into two subtasks. The goal of the two subtasks is to predict the complexity of a predetermined word. The evaluation index of the task is the Pearson correlation coefficient. Our best performance system has Pearson correlation coefficients of 0.7434 and 0.8000 in the single-token subtask test set and the multi-token subtask test set, respectively.

## 1 Introduction and Background

Language and writing are the main ways we transmit knowledge and information. An accurate and efficient understanding of the meaning expressed in the text is of great significance to our learning and production. Vocabulary complexity and reading comprehension are inextricably linked, and overly complex terms may bring bad results (DuBay, 2004). The research of Leroy et al. showed that the use of vocabulary simplification technology is one of the ways to effectively improve readers' reading comprehension ability (Leroy et al., 2013). Accurately predicting lexical complexity can make the system better guide users to use appropriate text, or customize text according to their needs. Especially when some ordinary readers are reading technical text content (Wei et al., 2009). Lexical complexity detection and complex lexical simplification have attracted the attention of the NLP community, and

systems have been developed to simplify the text of second language learners (Shardlow, 2014), native speakers with low literacy levels (Specia, 2010), and people with dyslexia (Rello et al., 2013).

The topic of the shared task of SemEval 2021 Task 1 is "Lexical Complexity Prediction (LCP)". The task data set uses English text data in a single language (Shardlow et al., 2020). There are two subtasks in the task, which are the subtasks for predicting the complexity of a single token and multiple tokens (Shardlow et al., 2021). In this article, we give the method and task result of predicting word complexity. Our system uses a method that combines sentence, word frequency, and context information. The acquisition of sentences and context information uses the RoBERTa model (Liu et al., 2019b). The word frequency information comes from the Tf-Idf algorithm (Ramos et al., 2003). The complexity is a continuous value, so the whole task can be regarded as a regression task. We provide the model code used in this task <sup>1</sup>.

## 2 Related Work

Previously held similar to this shared task are SemEval 2016 task 11: Complex Word Identification (CWI2016) (Paetzold and Specia, 2016a), Complex Word Identification Shared Task 2018 (CWI2018) (Yimam et al., 2018).

In CWI2016, the system voting method used by Paetzold et al. has achieved excellent results in sharing tasks (Paetzold and Specia, 2016b). In CWI2018, Butnaru uses a kernel-based learning method for complex word identification (Butnaru and Ionescu, 2018). Sian and other methods using integrated voting have also achieved good scores (Gooding and Kochmar, 2018). In addition to the above methods, some common methods are applied to these tasks. For example, SVM, random forest,

<sup>1</sup><https://github.com/Hub-Lucas/task1>



ID	corpus	sentence	token	complexity
subtask1-01	bible	He raises his hands against his friends.	hands	0.0278
subtask1-02	europarl	The first is Johann Wolfgang von Goethe.	Goethe	0.5
subtask2-01	bible	You shall tread on their high places..	high places	0.1625
subtask2-02	europarl	The first is legislative efficacy.	legislative efficacy	0.3833

Table 1: The training set sample data we use in the task. Subtask 1 has only one token, and subtask 2 has two tokens.

ture. The structure of the Inception Block used in our system is an improvement based on the solution implemented by Szegedy et al. (Szegedy et al., 2015). In the Inception block, we use the Conv1d convolution provided by Pytorch to adapt to our needs in the task. Inception Block has convolution kernels of different sizes, and can use windows of different sizes to extract non-continuous semantic features. At the same time, the parallel structure of Inception Block can save training time. The structure of the transformer allows interaction between the input sentence and the input token. We use the output of RoBERTa and the output of Tf-Idf weighted RoBERTa as different inputs of Inception Block, so that Inception Block can capture different information. Different classifiers are used to process the output results from different inputs of the Inception Block.

In step 1, we spliced the text data (Sentence) and the target word (Token) in the data with (SEP). Then the spliced result is used as the input of RoBERTa and Tf-Idf. In step 2, we use the output of Tf-Idf to weight the output of RoBERTa. In step 3, we use the weighted result of the previous step and the output result of RoBERTa as the input of the Inception Block. Here, the Inception Block is used as a shared layer to learn the output results of RoBERTa and the output results of RoBERTa weighted by Tf-Idf. In step 4, two linear classifiers are used to process the output from the Inception Block. In step 5, the output results of the two linear classifiers are averaged. In step 6, the average value is output as the final prediction result of the system.

## 4 Experiment and Results

In this section, we will introduce the data preprocessing methods and experimental settings we used in the task and the final results.

### 4.1 Data Preprocessing

In the part of data processing, we deleted the stop words in the text data. For the stop word list, we

use the stop word package provided by NLTK. To use the Tf-Idf algorithm to obtain a weighted output, and to ensure that the shape of the text code processed by the Tf-Idf algorithm is consistent with the shape of the RoBERTa output, we removed the part of the text code that exceeded the maximum sentence. For those text encodings that are less than the maximum sentence length, we perform zero padding. The encoding of Tf-Idf is obtained using the toolkit provided by gsim (Řehůřek and Sojka, 2010)<sup>2</sup>. For the validation set, we randomly select 20% from the pre-processed training set as our validation set during the training process. The remaining 80% of the training set is used as our training set during the training process.

### 4.2 Experiment setting

During our training model, we designed 4 different models and observed the result scores of different models on the validation set. We adjust the parameters as much as possible to obtain the best results for each different model, so different systems may have different parameter combinations. The overall design and data flow of the BERT+Tf-Idf+Inception system is the same as the system we introduced in Figure 2. The difference is that we replace the RoBERTa model in Figure 2 with the BERT model. In all experiments, we use Radam (Liu et al., 2019a) as the optimizer and MSELoss as the loss function.

- RoBERTa+Tf-Idf+CNN: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 60, and 4e-5, respectively.
- BERT+Tf-Idf+CNN: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 60, and 3e-5, respectively.
- RoBERTa: The epoch, batch size, maximum

<sup>2</sup><https://github.com/RaRe-Technologies/gensim>

Team	subtask	Pearson	Spearman	MAE	MSE	R2
Top1	1	0.7886	0.7369	0.0609	0.0062	0.6172
Top2	1	0.7882	0.7425	0.0610	0.0061	0.6210
Top3	1	0.7790	0.7355	0.0619	0.0064	0.6062
Our	1	0.7434	0.6995	0.0658	0.0073	0.5486
Top1	2	0.8612	0.8526	0.0616	0.0063	0.7389
Top2	2	0.8575	0.8529	0.0672	0.0072	0.7035
Top3	2	0.8571	0.8548	0.0675	0.0072	0.7012
Our	2	0.8000	0.7797	0.0754	0.0089	0.6323

Table 2: The scores of the top three teams and our team on the test set announced by the task organizer. Mean absolute error (MAE), Mean squared error (MSE), R-squared (R2). 61 and 38 different teams submitted results for subtask 1 and subtask 2, respectively.

sequence length, and learning rate for the model are 4, 32, 60, and 3e-5, respectively.

- BERT: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 60, and 3e-5, respectively.

## 5 Results and Analysis

According to the Pearson correlation coefficient, the results submitted by the teams participating in the two subtasks are ranked. In the published results, the task organizer team also announced some other evaluation indicators. These evaluation indicators are Spearman correlation (Rho), Mean absolute error (MAE), Mean squared error (MSE), R-squared (R2). We compare the scores of Pearson correlation coefficient results obtained by several different methods proposed in the experimental part. The results of these different methods can be found in Table 3.

Compare our result scores on the validation set of the two subtasks. First of all, our system can predict the word complexity required in the task. Secondly, under the same data and parameters, the score obtained by the RoBERTa model is higher than the score obtained by the BERT model. Then, the scores we get on the RoBERTa+Tf-Ifd+Inception and BERT+Tf-Ifd+Inception systems are higher than the single use of the RoBERTa model and the use of the BERT model. Finally, the above performance proves the feasibility of the improved method we used.

After comparing the result scores of different systems on the verification set, we used the RoBERTa+Tf-Ifd+Inception system to predict the results of the test set and successfully submitted it to the task organizer team. Our test set prediction result scores are 38th and 22nd respectively in the

Method	Pearson(1)	Pearson(2)
RoBERTa+Tf-Ifd+Inception	0.7651	0.8072
BERT+Tf-Ifd+Inception	0.7426	0.7850
RoBERTa	0.7327	0.7846
BERT	0.7255	0.7644

Table 3: The scores of the Pearson correlation coefficient results obtained by our different systems on the validation set. The validation set comes from 20% of the training set provided by the task organizer. Pearson(1) is the result score of the Pearson correlation coefficient of subtask 1. Pearson(2) is the result score of the Pearson correlation coefficient of subtask 2.

ranking lists of the two subtasks. Table 2 shows the test set result scores of the top three teams and our team announced by the task organizer team.

## 6 Conclusion

In this article, we describe the system our team has developed for shared tasks in SemEval 2021 task1 LCP. The system combines lexical, syntactic, and contextual semantic features. We describe and analyze the tasks, data, experiments, and results. We compared the results of the RoBERTa model and the BERT model. In the final test set prediction result score ranking, our results in the competition ranked middle. In future work, we will study how the complexity of the phrase is affected by the context in the sentence. For our model and method, we can also try to introduce other types of word embedding, and use different models to fuse the output of RoBERTa and the output of word embedding.

## References

- Andrei M Butnaru and Radu Tudor Ionescu. 2018. Unibuckernel: A kernel-based learning method for complex word identification.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- William H DuBay. 2004. The principles of readability. *Online Submission*.
- Sian Gooding and Ekaterina Kochmar. 2018. **CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting**. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Gondy Leroy, James E Endicott, David Kauchak, Obay Mouradi, and Melissa Just. 2013. User evaluation of the effects of a text simplification algorithm using term familiarity on perception, understanding, learning, and information retention. *Journal of medical Internet research*, 15(7):e144.
- Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019a. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Christopher Olah. 2015. Understanding lstm networks.
- Gustavo Paetzold and Lucia Specia. 2016a. **SemEval 2016 task 11: Complex word identification**. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Gustavo Paetzold and Lucia Specia. 2016b. Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. Simplify or help? text simplification strategies for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, pages 1–10.
- Matthew Shardlow. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. **CompLex — a new corpus for lexical complexity prediction from Likert Scale data**. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.
- Lucia Specia. 2010. Translating from complex to simplified sentences. In *International Conference on Computational Processing of the Portuguese Language*, pages 30–39. Springer.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Ruth Chung Wei, Linda Darling-Hammond, Alethea Andree, Nikole Richardson, and Stelios Orphanos. 2009. Professional learning in the learning profession: A status report on teacher development in the us and abroad. technical report. *National Staff Development Council*.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. **A report on the complex word identification shared task 2018**. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

# Manchester Metropolitan at SemEval-2021 Task 1: Convolutional Networks for Complex Word Identification

**Robert Flynn**

School of Computing, Mathematics  
and Digital Technology  
Manchester Metropolitan University  
robert.flynn@stu.mmu.ac.uk

**Matthew Shardlow**

School of Computing, Mathematics  
and Digital Technology  
Manchester Metropolitan University  
m.shardlow@mmu.ac.uk

## Abstract

We present two convolutional neural networks for predicting the complexity of words and phrases in context on a continuous scale. Both models utilize word and character embeddings alongside lexical features as inputs. Our system displays reasonable results with a Pearson correlation of 0.7754 on the task as a whole. We highlight the limitations of this method in properly assessing the context of the target text, and explore the effectiveness of both systems across a range of genres. Both models were submitted as part of LCP 2021, which focuses on the identification of complex words and phrases as a context dependent, regression based task.

## 1 Introduction

Complex Word Identification (CWI) involves identifying words that the reader may find difficult to understand. A word’s complexity can depend on many factors and differ according to context. Further, assessment of the complexity of named entities can require some degree of general knowledge, making CWI a challenging task (Shardlow, 2013). Accurately identifying complex words is important for many downstream simplification tasks, making literature more accessible for people with conditions such as dyslexia (Rello et al., 2013), and the assessment of a text’s readability as a whole (Dubay, 2004).

Our methodology plans to extend on previous convolutional network based approaches to CWI (Aroyehun et al., 2018; Sheang, 2019). With the goal of producing a system that is able to better model the complexities of phrases and unfamiliar words, within the English language.

Previous shared tasks on CWI addressed the problem as a binary and probabilistic classification type task, although human judgements on word complexity are not binary and exist on a continuous

scale. Lexical Complexity Prediction (LCP) 2021 tries to address this and uses an augmented version of CompLex (Shardlow et al., 2020), a dataset annotated with a 5-point Likert scale. CompLex also features context-specific annotation, with words receiving different annotations depending on their context. The dataset provides annotations from three different domains: Bible, Biomed and Europarl (Shardlow et al., 2021).

The code for this task is available on GitHub<sup>1</sup>.

## 2 Related Work

Word frequency is a commonly used feature in CWI (Gooding and Kochmar, 2018; Kajiwara and Komachi, 2018); words that appear frequently in language are more likely to be recognised and understood by the reader (Carroll et al., 1998). For the purpose of identifying medical terminology that may be unfamiliar to the lay reader, Elhadad (2006) leveraged lexical frequencies while also exploring the potential of other features such as word familiarity ratings from the MRC Psycholinguistic Database (Coltheart, 1981).

More recently lexical and psycholinguistic features have been utilized by machine learning tools, resulting in improved accuracy on these tasks. Through the use of an ensemble-based voting method the CAMB system (Gooding and Kochmar, 2018) achieved state-of-the-art results in the 2018 CWI shared task (Yimam et al., 2018), employing a total of 27 lexical, morphological and psycholinguistic features. The CAMB system however does not consider the target words context, opting for a “greedy” approach towards phrase classification, marking all phrases as complex.

Aroyehun et al. (2018) explored the use of convolutional neural networks (CNN) for CWI using only

<sup>1</sup><https://github.com/robflynnh/CNN-LCP-Shared-Task-2021>

the word embeddings of the target words and the averaged embeddings of the left and right contexts as inputs. They contrasted the results against a feature engineering approach using decision tree learning finding that both methods achieved competitive results. However, their decision tree method was marginally more accurate than their CNN for most of the datasets. Integrating lexical features alongside word embeddings can lead to further improvements in accuracy making this a more competitive approach, and outperforming many previous deep learning methods for CWI (Sheang, 2019).

By framing CWI as a sequence labelling task, Bi-directional long short-term memory (BiLSTM) networks have produced state-of-the-art results on the CWIG3G2 dataset (Yimam et al., 2017; Gooding and Kochmar, 2019). BiLSTM networks are able to capture long-term word and character level dependencies allowing these models to consider a large amount of contextual information. Modelling the complexity of phrases has proven to be a more challenging and complex task compared to individual words (Gooding and Kochmar, 2019).

### 3 Implementation

#### 3.1 Features

Below a description of the features used by both models is given:

**Frequency:** Word frequencies are taken from the SUBTLEX-UK word frequency database (van Heuven et al., 2014). Logarithmic Zipf frequency values were chosen based on previous results from this metric (Zampieri et al., 2016) and the Zipfian distribution that is displayed in language (Zipf, 1949).

**Age of Acquisition:** Age of Acquisition (AoA) values, estimating the age at which a word is typically acquired. (Kuperman et al., 2012; Brysbaert, 2012).

**Word-level Features:** Target word length and number of syllables are used as features (Brysbaert, 2012).

**Corpus Type:** As the dataset includes extracts from three different sources of potentially varying complexity, the corpus type was included and represented as a one-hot embedding.

**Pre-trained Embeddings:** 50d GloVe (Pennington et al., 2014) word embeddings, and 50d chars2vec<sup>2</sup> embeddings representing a word's char-

acter sequence are used. 50d GloVe embeddings were chosen as embeddings with more dimensions showed worse performance on the training data. Which suggests that the 50d embeddings capture sufficient information needed for this task. Character embeddings allow inferences to be made between words with similar morphologies.

#### 3.2 Preprocessing

Firstly min-max normalization is applied to the features taken from datasets, and word length is divided by 10. Non-alphanumeric characters are removed from the sentences before any features are extracted.

Both models take as inputs the features for the target word, and the averaged features for the left and right contexts of the target text. If the target word or words are positioned at the beginning or end of the sentence a zero vector of size 107 is used for the left or right context. For out-of-vocabulary words a zero vector is used for the word embedding and other features are imputed using mean values from their respective datasets. Finally the vectors for the target text and its context are stacked to produce a 3x107 matrix (left context — token — right context) for single words or a 4x107 matrix for MWEs (left context — token 1 — token 2 — right context).

#### 3.3 Models

This section provides a description of the architecture and hyperparameters used for both models. The models were produced using the Keras library version 2.4.3. Each of the models were trained with a batch size of 50, early stopping of 1000 and model checkpointing based on the validation loss.

##### 3.3.1 Single Word Model

For single words a 1D convolutional network followed by three fully connected layers is used. The model takes three inputs, an average of the features for left and right contexts is used for the first and third inputs respectively, and the features of the target word is used as the second input. The convolutional layer pads the inputs and uses a kernel size of 3 with 150 output filters and ReLu as the activation function. Global Max Pooling and a flatten layer followed by batch normalization is then applied to the output of this layer. Three dense layers with sizes of 150 (ReLu), 50 (ReLu) and 1 (Linear) are then used with a Dropout of 0.5 applied before each dense layer. Mean Squared Error

<sup>2</sup><https://github.com/IntuitionEngineeringTeam/chars2vec>



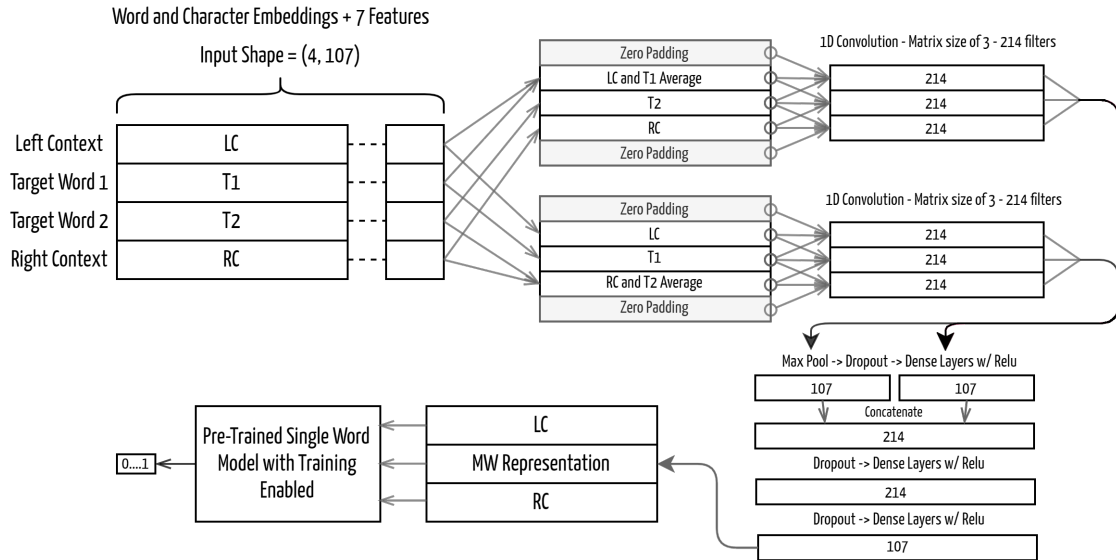


Figure 1: Depiction of multi-word model architecture

(MSE) is used as the loss function and Stochastic Gradient Descent as the optimizer, with a learning rate of 0.01 and momentum of 0.6 with Nesterov accelerated gradient enabled.

### 3.3.2 Multi-Word Model

For multi-words a second model is used to assess the complexity of two word phrases. This model acts as an adapter with the output being fed into a pre-trained single word model, allowing the model to take advantage of the data for single words and MWEs. Figure 1 gives an overview of the model architecture.

Features for the averaged left context, target word one, target word two and the averaged right context are used as input for the model. A convolutional layer with a similar architecture to task one is used for each of the target words. For the two convolutional layers the other target word is averaged with either the left or right context depending on its positioning, weighting the other target word higher than the rest of the context.

Each convolutional layer uses a filter size of 214 but is otherwise the same as in task one. Global Max Pooling followed by Dropouts of 0.3 and dense layers with 107 neurons and ReLu activation are applied to the outputs of the convolutions which are then concatenated along the last axis. Two dense layers with ReLu activation and sizes of 214 and 107 are then applied with a Dropout of 0.5 before each layer. This final output of size 107 is then concatenated along the first axis with the original left and right contexts to form the input

for a pre-trained single word model with training enabled. This model uses the Adam optimizer with default parameters and MSE as the loss function.

## 4 Results

Task	Pearson	MSE	R2
Task 1	0.7389	0.0074	0.5398
Task 2	0.7754	0.0079	0.5989

Table 1: Results for both tasks

This section will discuss and evaluate the performance of both models. Participants were ranked according to the Pearson correlation coefficient of their submissions. Table 1 presents the results for each of the tasks with task 1 evaluating individual words and task 2 covering both single and two word Multi-Word Expressions (MWEs).

### 4.1 Single Word Model Results

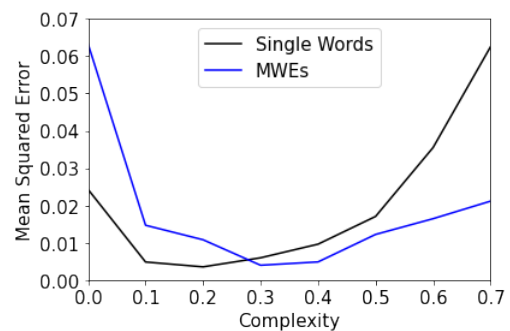


Figure 2: MSE across different complexities

As shown in Figure 2 the single word model struggles to accurately predict values for words of a high complexity, and also displays difficulties for words of a complexity of less than 0.1. The training and evaluation data features less examples of very simple or complex words. The complexity of these extremities is often highly dependant on the context, making them more challenging to assess.

Corpus	Pearson	MSE	R2
All	0.7389	0.0074	0.5398
Bible	0.7085	0.0085	0.4948
Biomed	0.7828	0.0087	0.6050
Europarl	0.6807	0.0055	0.4562
<b>JUST-BLUE</b>	0.7886	0.0062	0.6172

Table 2: Results for individual words

Table 2 presents the results for this task on each of the domains and the task as a whole. The prediction accuracy varies significantly across the different sources. Results from the best performing team are given for comparison (Shardlow et al., 2021).

As the model only uses an average of the features present in the left and right context of the target word, it is unable to differentiate between tokens that are influential to the target words complexity and ones that are not. Because of this equal weighting of words in the context, the models accuracy can be negatively affected by an abundance or lack of stop words in the sentence. Very complicated or simple words in sentence that are not related to the target word, and don't share a similar complexity can also cause the model to over- or under-predict the target word's complexity. The mechanism by which the model assesses the context may partly explain the variance in accuracy on each domain.

Interestingly, our sub-analysis showed that the model shows a better correlation for those tokens without a word embedding, yielding a Pearson correlation of 0.7804 and a MSE of 0.0071. Generally these out-of-vocabulary words are more complex so the model is using the lack of a word embedding as a feature when making predictions. Although this shows a better correlation overall it could lead to false positives in specific cases where the out-of-vocabulary word is of a low complexity.

## 4.2 Multi-Word Model Results

As shown in Figure 2 the multi-word model is much less accurate for very simple MWEs of a complexity less than 0.1. However, for more complex words

the predictions remain fairly accurate. This model is able to assess the way in which the words in a phrase interact with each other and to some degree the rest of the sentence. This additional contextual information may increase the model's capacity to evaluate more complex words. Only 1.65 percent of phrases in the training data were of a complexity of less than or equal to 0.1 which could explain the inaccuracy in this range.

Corpus	Pearson	MSE	R2
All	0.7611	0.0102	0.5770
Bible	0.7173	0.0113	0.5106
Biomed	0.7980	0.0141	0.6317
Europarl	0.5799	0.0060	0.3089
<b>DeepBlueAI</b>	0.8612	0.0063	0.7389

Table 3: Results for MWEs

MWE Type	Pearson	MSE	R2
A-N (115)	0.7654	0.0115	0.5801
N-N (56)	0.7414	0.0091	0.5293

Table 4: Results for the different MWE formations. A-N: Adjective-Noun. N-N: Noun-Noun.

Table 3 presents the results across each of the different domains present in the dataset. The model used for MWEs makes use of a fine-tuned instance of the single-word model; consequentially incorrect associations from the single-word model may have been carried over to this model. The results show a similar variance across domains to task 1, although it struggles more significantly on the Europarl sub-corpus. Compared to the other domains, Europarl's complexity values show a much smaller standard deviation than the other sub-corpora (0.093 compared to 0.196 and 0.152, on biomed and bible). The variation of complexities may play a role in the models effectiveness at making accurate predictions across the domains.

Table 4 presents the results across different MWE formations. The number of occurrences of each part-of-speech formation is denoted in brackets, MWE types with less than 10 occurrences were omitted from the table. The model performs marginally better on Adjective-Noun MWE formations.

## 5 Discussion

In this paper, we presented two convolutional neural networks used as an approach to single-word and multi-word complex word identification. Both models achieved reasonable results, achieving scores in a comparable range to the majority of other submissions.

Multi-Word CWI is a more challenging task compared to the assessment of single words; the multi-word model was able to utilize the datasets of both tasks, and its predictions show a Pearson's correlation score of 0.7611. Our system is only able to process two-word MWEs, which for this task is not an issue. However, in other use cases the ability to assess longer MWEs would be useful. Given a dataset with annotations for longer MWEs the model could potentially be adapted to work with three or four word sequences.

Both models are able to assess the context of the target text when making predictions; although, as the left and right contexts are given as an average, all words are weighted equally regardless of their relevance to the target text. Because of this equal weighting of words, the models are able to adjust their predictions based on the general complexity of the sentence but are unable to fully capture the relevant context. Adding a mechanism that could weight each word in the context based on certain features may offer some improvements in this area. Attention based models such as BERT (Devlin et al., 2019) are able to attend to each token in a sequence to produce embeddings that capture large amounts of contextual information. Fine-tuning such a model on CWI tasks could produce embeddings that contain more useful and relevant information.

## References

Segun Taofeek Aroyehun, Jason Angel, Daniel Alejandro Pérez Alvarez, and Alexander Gelbukh. 2018. [Complex word identification: Convolutional neural network vs. feature engineering](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 322–327, New Orleans, Louisiana. Association for Computational Linguistics.

Marc Brysbaert. 2012. [crr ” age-of-acquisition \(aoa\) norms for over 50 thousand english words](#).

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic

readers. *Proc. of AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*.

- Max Coltheart. 1981. [The mrc psycholinguistic database](#). *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- William Dubay. 2004. The principles of readability. *CA*, 92627949:631–3309.
- Noemie Elhadad. 2006. [Comprehending technical texts: predicting and defining unfamiliar terms](#). *AMIA ... Annual Symposium proceedings. AMIA Symposium*, 2006:239–243. 17238339[pmid].
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2019. [Complex word identification as a sequence labelling task](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153, Florence, Italy. Association for Computational Linguistics.
- Walter van Heuven, Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. [Subtlex-uk: a new and improved word frequency database for british english](#). *Quarterly journal of experimental psychology (2006)*, 67.
- Tomoyuki Kajiwara and Mamoru Komachi. 2018. [Complex word identification based on frequency in a learner corpus](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 195–199, New Orleans, Louisiana. Association for Computational Linguistics.
- Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. [Age-of-acquisition ratings for 30,000 english words](#). *Behavior Research Methods*, 44(4):978–990.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

- Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013. [Simplify or help? text simplification strategies for people with dyslexia](#). In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility, W4A '13*, New York, NY, USA. Association for Computing Machinery.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.
- Kim Cheng Sheang. 2019. [Multilingual complex word identification: Convolutional neural networks with morphological and linguistic features](#). In *Proceedings of the Student Research Workshop Associated with RANLP 2019*, pages 83–89, Varna, Bulgaria. INCOMA Ltd.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.
- Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. [CWIG3G2 - complex word identification task across three text genres and two user groups](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.
- Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. [MacSaar at SemEval-2016 task 11: Zipfian and character features for ComplexWord identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1001–1005, San Diego, California. Association for Computational Linguistics.
- George K. Zipf. 1949. *Human Behavior and the Principle of Least Effort*.

# UPB at SemEval-2021 Task 1: Combining Deep Learning and Hand-Crafted Features for Lexical Complexity Prediction

George-Eduard Zaharia, Dumitru-Clementin Cercel, Mihai Dascalu

University Politehnica of Bucharest, Faculty of Automatic Control and Computers

george.zaharia0806@stud.acs.upb.ro  
{dumitru.cercel, mihai.dascalu}@upb.ro

## Abstract

Reading is a complex process which requires proper understanding of texts in order to create coherent mental representations. However, comprehension problems may arise due to hard-to-understand sections, which can prove troublesome for readers, while accounting for their specific language skills. As such, steps towards simplifying these sections can be performed, by accurately identifying and evaluating difficult structures. In this paper, we describe our approach for the SemEval-2021 Task 1: Lexical Complexity Prediction competition that consists of a mixture of advanced NLP techniques, namely Transformer-based language models, pre-trained word embeddings, Graph Convolutional Networks, Capsule Networks, as well as a series of hand-crafted textual complexity features. Our models are applicable on both subtasks and achieve good performance results, with a MAE below 0.07 and a Person correlation of .73 for single word identification, as well as a MAE below 0.08 and a Person correlation of .79 for multiple word targets. Our results are just 5.46% and 6.5% lower than the top scores obtained in the competition on the first and the second subtasks, respectively.

## 1 Introduction

Reading is a complex process due to the mental exercise readers are challenged to perform, since a coherent representation of the text needs to be projected into their mind in order to grasp the underlying content (Van den Broek, 2010). For non-native speakers, the lack of text understanding hinders knowledge assimilation, thus becoming the main obstacle that readers need to overcome. Complex words can impose serious difficulties, considering that their meaning is often dependant on their context and cannot be easily inferred. In order to facilitate text understanding or to perform text simplification, complex tokens first need to be detected.

This can be performed by developing systems capable of identifying them by individual analysis, as well as contextual analysis.

There are two main approaches regarding the complexity task. Tokens can be binary classified as complex or non-complex, a procedure that helps users separate problematic words from the others. Words can be also labeled with a probabilistic complexity value, which in return can be used to simplify the text. Words with lower degrees of complexity can be easily explained, whereas more complex tokens can be replaced with simpler equivalent concepts.

The Lexical Complexity Prediction (LCP) shared task, organized as the SemEval-2021 Task 1 (Shardlow et al., 2021a), challenged the research community to develop robust systems that identify the complexity of a token, given its context. Systems were required to be easily adaptable, considering that the dataset entries originated from multiple domains. At the same time, the target structure evaluated in terms of complexity could contain a single word or multiple words, depending on the subtask.

The current work is structured as follows. The next section presents the state-of-the-art Natural Language Processing (NLP) approaches for LCP (probabilistic) and complex word identification (CWI). The third section outlines our approaches for this challenge, while the fourth section presents the results. Afterwards, the final section draws the conclusions and includes potential solutions that can be used to further improve performance.

## 2 Related Work

**Probabilistic CWI.** Kajiwara and Komachi (2018) adopted for the CWI task a system based on Random Forest regressors, alongside several features, such as the presence of the target word in certain

corpora. Moreover, they conducted experiments to determine the best parameters for their regression algorithms.

De Hertog and Tack (2018) introduced a deep learning architecture for probabilistic CWI. Apart from the features extracted by the first layers of the network, the authors also included a series of hand-crafted features, such as psychological measures or frequency counts. Their architecture included different Long Short-Term Memory (LSTM) modules (Hochreiter and Schmidhuber, 1997) for the input levels (i.e., word, sentence), as well as the previously mentioned psychological measures and corpus counts.

**Sequence labeling CWI.** Gooding and Kochmar (2019) introduced a technique based on LSTMs for CWI, which obtained better results on their sequence labeling task than previous approaches based only on feature engineering. The contexts detected by the LSTM offered valuable information, useful for identifying complex tokens placed in sequences.

Changing the focus towards text analysis, Finnimore et al. (2019) extracted a series of relevant features that supports the detection of complex words. While considering their feature analysis process, the greatest influence on the overall system performance was achieved by the number of syllables and the number of punctuation marks accompanying the targeted tokens.

A different approach regarding CWI was adopted by Zampieri et al. (2017), who employed the usage of an ensemble created on the top systems from the SemEval CWI 2016 competition (Paetzold and Specia, 2016). Other experiments performed by the authors also included plurality voting (Polikar, 2006), or a technique named Oracle (Kuncheva et al., 2001), that forced label assignment only when at least one classifier detected the correct label.

Zaharia et al. (2020) tackled CWI through a cross-lingual approach. Resource scarcity is simulated by training on a small number of examples from a language and testing on different languages, through zero-shot, one-shot, and few-shot scenarios. Transformer-based models (Vaswani et al., 2017) achieved good performance on the target languages, even though the number of training entries was extremely reduced.

## 3 Method

### 3.1 Dataset

CompLex (Shardlow et al., 2020, 2021b) is the dataset used for the LCP shared task that was initially annotated on a 5-point Likert scale. Moreover, the authors performed a mapping between the annotations and values between 0 and 1 in order to ensure normalization. The dataset has two categories, one developed for single word complexity score prediction, while the other is centered on groups of words; each category has entries for training, trial, and testing. The single word dataset contains 7,662 entries for training, 421 trial entries, and 917 test entries. The multi-word dataset contains a considerably smaller number of entries for each category, namely 1,517 for training, 99 trial entries, and 184 for testing.

All entries from the LCP shared task are part of one of three different English corpora (i.e., Bible - biblical entries, Biomed - biomedical entries, and Europarl - political entries), evenly distributed, each one representing approximately 33% of its corresponding set. As such, the task is even more challenging when considering the vastly different domains of these entries.

### 3.2 Architecture

During our experiments, we combined features obtained from multiple modules described later on in detail, and then applied three regression layers, alongside a *Dropout* layer, to obtain the complexity score of the input (see Figure 1 for our modular architecture). The permanent components are represented by the target word embeddings and the Transformer features, which are concatenated and then fed into the final linear layers, designated for regression. The other components (i.e., character-level embeddings, GCN, and Capsule) are enabled in particular setups; similarly, the adversarial training component can also be disabled. At the same time, a series of hand-crafted features can be concatenated before the last layer with the aim to further improve the overall performance.

### 3.3 Pre-trained Word Embeddings

Pre-trained word embeddings were used as features for the final regression as an initial representation of the input. Throughout our experiments, three types of pre-trained word embeddings were consid-

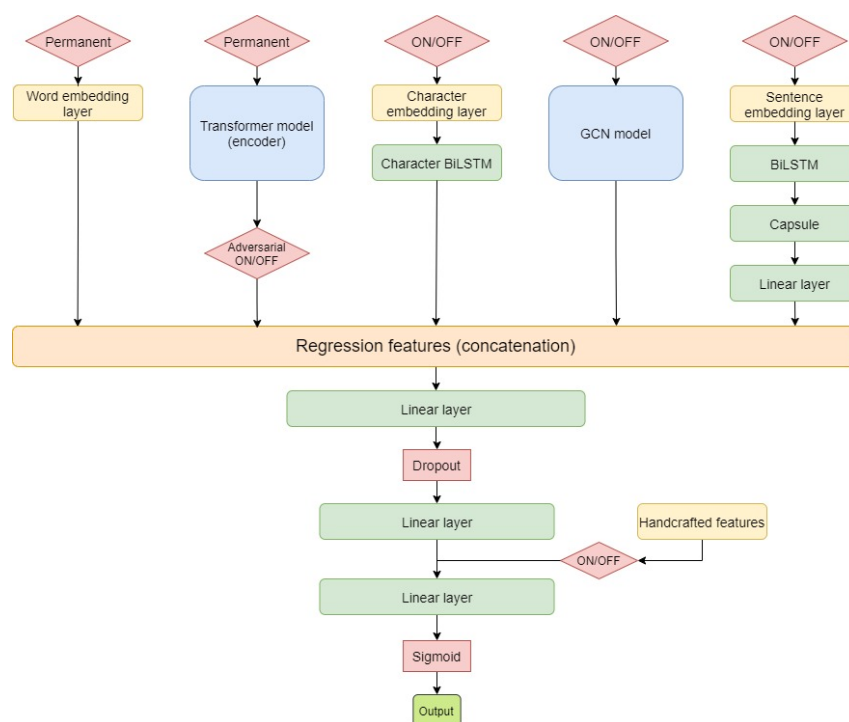


Figure 1: The overall model architecture used in our experiments.

ered, namely: GloVe<sup>1</sup>, FastText<sup>2</sup>, and skip-gram<sup>3</sup>. Out of the three previous options, GloVe performed best in our experiments. As such, the results section exclusively reports the performance obtained by our configurations alongside GloVe embeddings for the target word.

### 3.4 Transformer-based Language Models

Considering that Transformers achieve state-of-the-art performance for most NLP tasks (Wolf et al., 2020), all our setups include a Transformer-based component. However, they are pre-trained in different manners; thus, we experimented with several variants, as follows:

- **BERT** (Devlin et al., 2019) - Extensively pre-trained on English, BERT-base represents the baseline of Transformer-based models;
- **BioBERT** (Lee et al., 2020) - Considering that some of the most difficult to understand entries are part of the Biomed corpus, we also experimented with a model pre-trained on biomedical data;
- **SciBERT** (Beltagy et al., 2019) - Similarly to BioBERT, SciBERT is pre-trained on scien-

tific data and becomes a good candidate for fine-tuning on the scientific entries from the dataset;

- **RoBERTa** (Liu et al., 2019) - RoBERTa improves upon BERT by modifying key hyperparameters, and by being trained with larger mini-batches and learning rates; RoBERTa usually has better performance on downstream tasks.

### 3.5 Adversarial Training

We also aimed to improve the robustness of the main element of our architecture, the Transformer-based component. Therefore, we adopted an adversarial training technique, similar to Karimi et al. (2020). The adversarial examples generated during training work on the embeddings level, and are based on a technique that uses the gradient of the loss function.

### 3.6 Character-level Embeddings

Alongside the previously mentioned word embeddings for the target word, we also employ character-level embeddings for the same word, such that its internal structure, as well as its universal context, can be properly captured as features in our architecture.

<sup>1</sup><https://nlp.stanford.edu/projects/glove/>

<sup>2</sup><https://fasttext.cc/>

<sup>3</sup><http://vectors.nlpl.eu/repository/>

### 3.7 Graph Convolutional Networks

Besides the previous Transformer-based models, we also explored the relations between the dataset entries, as well as the vocabulary words. Therefore, Graph Convolutional Networks (GCN) (Kipf and Welling, 2016) were also considered for determining node embedding vectors, by taking into account the properties of the neighboring nodes. By stacking multiple GCN layers, the information embedded into a node can become broader, inasmuch as it incorporates considerably larger neighborhoods. Similar to Yao et al. (2019), we consider the graph to have several nodes equal to the number of entries (documents) in the corpus plus the vocabulary size of the corpus.

### 3.8 Capsule Networks

Alongside the relational approach derived from GCN and Transformer embeddings, we intended to further analyze our inputs by passing them through a Capsule Network (Sabour et al., 2017). This approach enables us to obtain features that reflect aspects specific to different levels of the inputs, as Capsule Networks increase the specificity of features, while the capsule layers go deeper.

### 3.9 Hand-crafted Features

Similar to Gooding and Kochmar (2018), we integrated a series of hand-crafted features for the target word: **Syllables**, **Synset length**, **Hyponyms length**, **Hyponyms length**, **Number of dependencies** obtained using NLTK<sup>4</sup> alongside CoreNLP<sup>5</sup>, **SubIMDB presence**<sup>6</sup>, **SimpWiki presence** (Coster and Kauchak, 2011), **CEFR level** obtained from the Cambridge English dictionary<sup>7</sup>, **MRC features** (Wilson, 1988) (*Age of acquisition*, *Concreteness rating*, *Imageability rating*, *Word familiarity rating*, *Number of phonemes*), **Semantic Diversity** (Hoffman et al., 2013), **Sensorimotor Norms** (Lynott et al., 2019).

**Character n-grams** - The character n-gram approach consists of two steps: first, a vectorizer is applied on the inputs to select a maximum number of 5,000 most frequent n-grams; second, Tf-Idf scores for these elements are computed. The obtained values are then normalized in the  $[0, 1]$  range

<sup>4</sup><https://www.nltk.org/>

<sup>5</sup><https://stanfordnlp.github.io/CoreNLP/>

<sup>6</sup><http://ghpaetzold.github.io/subimdb/>

<sup>7</sup><https://dictionary.cambridge.org/dictionary/learner-english/>

and used as features.

**ReaderBench indices** - The ReaderBench framework (Dascalu et al., 2017) was used to extract additional textual complexity features reflective of writing style. Out of the 1311 features obtained by running ReaderBench on our inputs, we selected 278. The choice was made by considering only the features with a high linguistic coverage (i.e. were non-zero for at least 50% of the entries).

### 3.10 Traditional Machine Learning Baseline

Several machine learning algorithms, such as *Logistic regression*, *Random Forest Regressors*, *XGBoost regression*, or *Ridge regression* were experimented using the aforementioned handcrafted features.

We then switched to a ridge regression approach and trained it with a multitude of features, including Transformer-based embeddings (BERT, BioBERT, SciBERT, RoBERTa), pre-trained word embeddings (GloVe, fastText, Skip-gram), and handcrafted features.

### 3.11 Preprocessing and Experimental Setup

Text preprocessing is minimal and consists of removing unnecessary punctuation, such as quotes. The experimental hyperparameters for all modules are presented in Table 1.

## 4 Results

Table 2 introduces the results obtained using our deep learning architecture, while Table 3 focuses on the traditional machine learning baseline. The best results for the deep learning approaches applied on the single target word dataset are obtained using RoBERTa as Transformer model. The setup which maximizes performance considers RoBERTa, GCN, and Capsule features, obtaining a Pearson score of 0.7702 and a mean absolute error (MAE) of 0.0671 on the trial dataset. Moreover, the high performance is maintained on the test dataset, with a Pearson correlation coefficient of 0.7237 and a MAE of 0.0677. BERT, SciBERT, and BioBERT have similar results with marginal differences; GCN, Capsule, and adversarial training improve performance for all models, while character-level embeddings do not provide a boost in performance.

Table 3 presents the results obtained using the features described in Section 3, namely Transformer-based contextualized embeddings (BERT, RoBERTa, BioBERT, SciBERT), pre-



Transformers	GCN	Capsule	BiLSTM	Embedding	Full Model
Size: 768	GCN Size 1: 512 GCN Size 2: 256	Routings: 5 Number of capsules: 10 Capsule dimension: 16	Dimension: 128	Dimension: 300	Optimizer: AdamW Loss Function: MSELoss Learning Rate: $2e-5$

Table 1: Experimental hyperparameters for the probabilistic CWI.

Configuration	Single-Word Target				Multi-Word Target			
	Trial		Test		Trial		Test	
	Pearson	MAE	Pearson	MAE	Pearson	MAE	Pearson	MAE
BERT	0.7575	0.0689	0.7170	0.0682	0.7019	0.0969	0.7853	0.0781
BERT + Capsule	0.7641	0.0685	0.7113	0.0689	0.7170	0.0958	0.7774	0.0791
BERT + GCN + Capsule	0.7548	0.0693	0.7178	0.0682	0.6978	0.0905	0.7773	0.0812
BERT + GCN + Capsule + Adversarial Training	0.7608	0.0695	0.7171	0.0684	0.7077	0.0933	0.8008	0.0779
BERT + Char Embeddings	0.7505	0.0701	0.6925	0.0717	0.7091	0.0904	0.7800	0.0821
RoBERTa	0.7676	0.0685	0.7222	0.0681	0.7177	0.0925	0.7921	0.0764
RoBERTa + GCN + Capsule*	<b>0.7702</b>	<b>0.0671</b>	0.7237	<b>0.0677</b>	0.7160	0.0910	<b>0.7962</b>	0.0788
RoBERTa + GCN + Capsule + Adversarial Training*	0.7699	0.0682	<b>0.7324</b>	0.0703	<b>0.7227</b>	0.0893	0.7851	0.0808
RoBERTa + Hand-crafted Features	0.7476	0.0704	0.7028	0.0735	0.7165	0.0974	0.7932	<b>0.0754</b>
RoBERTa + Char Embeddings + Capsule + GCN + Adversarial	0.7663	0.0696	0.7264	0.0692	0.7221	0.0954	0.7958	0.0791
RoBERTa + Char Embeddings	0.7658	0.0695	0.7259	0.0682	0.7167	0.0958	0.7916	0.0772
SciBERT + GCN	0.7626	0.0714	0.7145	0.0715	0.6829	0.0876	0.7888	0.0762
SciBERT + GCN + Capsule + Adversarial Training	0.7617	0.0721	0.7086	0.0724	0.7164	<b>0.0863</b>	0.7882	0.0785
SciBERT + Char Embeddings	0.7512	0.0710	0.7079	0.0691	0.6855	0.1046	0.7729	0.0809
BioBERT	0.7658	0.0694	0.7151	0.0698	0.7014	0.0906	0.7814	0.0827
BioBERT + GCN + Capsule + Adversarial Training	0.7683	0.0677	0.7144	0.0690	0.7098	0.0968	0.7919	0.0795
BioBERT + Char Embeddings	0.7619	0.0689	0.7073	0.0697	0.7069	0.0995	0.7849	0.0810

\* The models marked with \* are the ones used in our submissions.

Table 2: Results for the Deep Learning approaches.

trained word embeddings (GloVe, fastText, skip-gram), and hand-crafted features, all combined using various regression algorithms. Logistic regression, Random Forrest and XGBosst yield lower performance when compared to the previous deep learning approaches. However, we managed to increase the scores on the single target word dataset, with Pearson coefficients of 0.7738 and 0.7340 on the trial and test datasets, by combining the results obtained from training several instances of ridge regression. Nevertheless, the best results for the multiple target word task are still obtained by the deep learning approaches (RoBERTa, GCN, Capsule, adversarial training), which surpass the Ridge Regression + pre-trained word embeddings + Transformer embeddings + handcrafted features approach by a low margin of 0.0074 Pearson on the trial dataset and 0.0033 on the test dataset.

## 5 Discussion

The entries with the largest difference when compared to the gold standard are represented by the ones that are part of the Biomed category. This discrepancy is valid for both subtasks (i.e, single target word and multiple target words). The Biomed entries employ the usage of more complex terminology, quantities, or specific scientific names. Therefore, it becomes more difficult for standard pre-trained Transformer systems, such as BERT or RoBERTa, to adapt to the Biomed entries. In contrast, corpora with easier to understand language (i.e., Bible and Europarl) are not properly represented when using BioBERT or SciBERT, considering that the Transformers are mainly pre-trained for scientific or biomedical texts.

Moreover, a considerable part of the Biomed entries contains large amounts of abbreviations, while other entries from the same domain have

Method	Single-Word Target				Multi-Word Target			
	Trial		Test		Trial		Test	
	Pearson	MAE	Pearson	MAE	Pearson	MAE	Pearson	MAE
Logistic Regression	0.7158	0.0748	0.6868	0.0718	0.6533	0.0954	0.7558	0.0791
Random Forest Regressor	0.7390	0.0708	0.7011	<b>0.0691</b>	0.6714	0.0929	0.7651	<b>0.0785</b>
XGBoost Regressor	0.7488	0.0700	0.7033	0.0695	0.6503	0.0975	0.7544	0.0804
Ridge Regression*	<b>0.7738</b>	<b>0.0686</b>	<b>0.7340</b>	0.0699	<b>0.7153</b>	<b>0.0873</b>	<b>0.7929</b>	0.0787

\* The solution marked with \* is the one used in our submissions.

Table 3: Results for the Traditional Machine Learning solutions.

Entry	Target	Predicted complexity	True complexity
Genetic analyses of sitosterolemia pedigrees allowed the mapping of the STSL locus to human chromosome 2p21, between D2S2294 and D2S2298 [12,13].	pedigrees	0.4516	0.3125
Normally cells accumulate H3-2meK9 and H3-3meK9 marks and HPIB protein on the sex chromatin as they progress to diplotene, but we observed mutant diplotene cells lacking these features.	marks	0.2686	0.3409
p150CAF-1 knockdown in ES cells was quantified by Western blot analysis and IF.	ES	0.5587	0.6944

Table 4: Difficult Biomed entries.

specific terms or links, as seen in Table 4. The difference between our predictions and the correct labels are up to 0.14 for the complexity probability.

## 6 Conclusions and Future Work

This work proposes a modular architecture, as well as different training techniques for the Lexical Complexity Prediction shared task. We experimented with different variations of the previously mentioned architecture, as well as textual features alongside machine learning algorithms. First, we used different word embeddings and Transformer-based models as the main feature extractors and, at the same time, we examined a different training technique based on adversarial examples. Second, other different models were added, such as character-level embeddings, Graph Convolutional Networks, and Capsule Networks. Third, several hand-crafted features were also considered to create a solid baseline covering both deep learning and traditional machine learning regressors.

For future work, we intend to experiment with altering the modular architecture such that the models are trained similar to a Generative Adversarial Network (Croce et al., 2020), thus further improving robustness and achieving higher scores in terms of both Pearson correlation coefficients and MAE.

## References

- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3606–3611.
- Paul Van den Broek. 2010. Using texts in science education: Cognitive processes and knowledge representation. *Science*, 328(5977):453–456.
- William Coster and David Kauchak. 2011. Learning to simplify sentences using Wikipedia. In *Proceedings of the workshop on monolingual text-to-text generation*, pages 1–9.
- Danilo Croce, Giuseppe Castellucci, and Roberto Basili. 2020. GAN-BERT: Generative adversarial learning for robust text classification with a bunch of labeled examples. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2114–2119.
- Mihai Dascalu, Gabriel Gutu, Stefan Ruseti, Ionut Cristian Paraschiv, Philippe Dessus, Danielle S McNamara, Scott A Crossley, and Stefan Trausan-Matu. 2017. ReaderBench: a multi-lingual framework for analyzing text complexity. In *European Conference on Technology Enhanced Learning*, pages 495–499. Springer.
- Dirk De Hertog and Anaïs Tack. 2018. Deep Learning Architecture for Complex Word Identification. In *Thirteenth Workshop of Innovative Use of NLP for Building Educational Applications*, pages 328–334. Association for Computational Linguistics (ACL); New Orleans, Louisiana.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Pierre Finnimore, Elisabeth Fritzsich, Daniel King, Alison Sneyd, Aneeq Ur Rehman, Fernando Alva-Manchego, and Andreas Vlachos. 2019. Strong Baselines for Complex Word Identification across Multiple Languages. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 970–977.
- Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- Sian Gooding and Ekaterina Kochmar. 2019. Complex word identification as a sequence labelling task. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Paul Hoffman, Matthew A Lambon Ralph, and Timothy T Rogers. 2013. Semantic diversity: A measure of semantic ambiguity based on variability in the contextual usage of words. *Behavior research methods*, 45(3):718–730.
- Tomoyuki Kajiwara and Mamoru Komachi. 2018. Complex word identification based on frequency in a learner corpus. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 195–199.
- Akbar Karimi, Leonardo Rossi, Andrea Prati, and Katharina Full. 2020. Adversarial training for aspect-based sentiment analysis with BERT. *arXiv preprint arXiv:2001.11316*.
- Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*.
- Ludmila I Kuncheva, James C Bezdek, and Robert PW Duin. 2001. Decision templates for multiple classifier fusion: an experimental comparison. *Pattern recognition*, 34(2):299–314.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
- Dermot Lynott, Louise Connell, Marc Brysbaert, James Brand, and James Carney. 2019. The Lancaster Sensorimotor Norms: multidimensional measures of perceptual and action strength for 40,000 English words. *Behavior Research Methods*, pages 1–21.
- Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Robi Polikar. 2006. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45.
- Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic Routing between Capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3859–3869.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting Lexical Complexity in English Texts. *arXiv preprint arXiv:2102.08773*.
- Matthew Shardlow, Marcos Zampieri, and Michael Cooper. 2020. CompLex—A New Corpus for Lexical Complexity Prediction from LikertScale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior research methods, instruments, & computers*, 20(1):6–10.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

- Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph Convolutional Networks for Text Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.
- George-Eduard Zaharia, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. Cross-Lingual Transfer Learning for Complex Word Identification. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 384–390. IEEE.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in Data Annotation and System Performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*, pages 59–63.

# UTFPR at SemEval-2021 Task 1: Complexity Prediction by Combining BERT Vectors and Classic Features

Gustavo Henrique Paetzold

Universidade Tecnológica Federal do Paraná - Campus Toledo

Toledo-PR, Brazil

ghpaetzold@utfpr.edu.br

## Abstract

We describe the UTFPR systems submitted to the Lexical Complexity Prediction shared task of SemEval 2021. They perform complexity prediction by combining classic features, such as word frequency, n-gram frequency, word length, and number of senses, with BERT vectors. We test numerous feature combinations and machine learning models in our experiments and find that BERT vectors, even if not optimized for the task at hand, are a great complement to classic features. We also find that employing the principle of compositionality can potentially help in phrase complexity prediction. Our systems place 45th out of 55 for single words and 29th out of 38 for phrases.

## 1 Introduction

Accurately measuring the complexity of words can be useful in many ways. It facilitates the creation of text simplification technologies that could, for example, help in identifying and adapting challenging excerpts of literary pieces targeting specific groups, such as children (De Belder and Moens, 2010) and second language learners (Paetzold and Specia, 2016e), and make news articles and official documents more accessible to the general population (Paetzold and Specia, 2016a).

This task has received a considerable amount of attention in the past few years, especially due to the popularity of the Complex Word Identification (CWI) shared tasks of 2016 (Paetzold and Specia, 2016c) and 2018 (Yimam et al., 2018), where dozens of teams were challenged to judge the complexity of words in context. While the CWI 2016 task used a simple binary complex/not complex classification setup for English only, the CWI 2018 task explored both a binary classification and a regression setup and multiple languages.

The majority of the most successful systems submitted to these shared tasks combined ensemble

methods, such as Random Forests (Ho, 1995) and AdaBoost (Freund and Schapire, 1997) with numerous linguistic features, including word frequencies, n-gram frequencies, word length, number of senses, number of syllables, psycholinguistic metrics, and word embeddings (Konkol, 2016; Malmasi et al., 2016; Paetzold and Specia, 2016d; Gooding and Kochmar, 2018; Hartmann and Dos Santos, 2018). However, because these tasks were held prior to the ascension of transformer-based masked language models, such as BERT (Devlin et al., 2019) and ROBERTA (Liu et al., 2019), we could not find any systems that exploited the power of the features produced by them.

In this paper, we describe the UTFPR systems for the Lexical Complexity Prediction shared task of SemEval 2021 (LCP 2021), which combine classic complexity prediction features with contextual word and phrase representations extracted from transformer-based models. In our experiments, we explore the efficacy of a number of different machine learning models, feature combinations, and corpora sources for our features. In what follows, we present the task being addressed (Section 2), our approach (Section 3), some preliminary experiments (Section 4), our final shared task results (Section 5), and our conclusions (Section 6).

## 2 Task Description

We address the LCP 2021 shared task (Shardlow et al., 2021), held at SemEval 2021. The shared task is split into two sub-tasks: predicting the in-context lexical complexity of single words and phrases for the English language. Participants could choose to submit systems to either or both sub-tasks.

The organizers provided training, trial and test sets for both sub-tasks. Each instance of these datasets is composed of an ID, a source identifier,

a sentence, a target word or phrase within the sentence, and a complexity score calculated based on judgments made by 20 English speakers from the USA, UK and Australia. The source identifier describes from where the sentence came from, the possibilities being the Bible, biomedical documents and the Europarl corpus. The task’s dataset is an extended version of the CompLex dataset (Shardlow et al., 2020).

The training, trial, and test sets for single words have 7662, 421, and 917 instances, respectively. The training, trial and test sets for phrases have 1517, 99, and 184 instances, respectively. Participants were allowed and encouraged to use any external resources they saw fit.

### 3 Approach

Our approach consists of using modern ensemble models to learn from a combination of commonly used complexity estimation features, such as word frequencies, word length, and number of senses, with contextual representations extracted from large pre-trained BERT-like models, which have been widely used to create state-of-the-art solutions to numerous tasks. While it has been observed that word frequencies (especially those extracted from spoken text) tend to drive the performance of effective complexity prediction systems (Paetzold and Specia, 2016c), we hypothesize that the wealth of knowledge present in transformer-based models such as BERT can help in extracting complementary contextual complexity clues.

#### 3.1 Features

We explore a set of 779 total features in our approach. They are:

- **Frequency:** We use not only word/phrase frequency, but also n-gram frequencies as well. We consider a total of 9 configurations  $(i, j)$ , where  $i$  represents the number of tokens to the left of the target word/phrase to be considered and  $j$  the number of tokens to the right. The configurations we consider are (0, 0), (0, 1), (1, 0), (1, 1), (0, 2), (2, 0), (1, 2), (2, 1), (2, 2).
- **Length:** We use the number of characters that compose the word/phrase. For phrases, instead of using its overall length, we use the average number of characters of all individual words. We motivate this decision in the experiments of Section 4.2.
- **Number of senses:** We use the word/phrase’s number of senses catalogued in the WordNet database (Miller et al., 1990). In line with our setup for word length, for phrases, we use the average number of senses of all individual words.
- **BERT vector:** We use the numerical representation of 768 dimensions produced by the pre-trained BERT model (Devlin et al., 2019). For phrases and out-of-vocabulary words that were fragmented during tokenization, we average the representations produced for all fragments. More specifically, we used the bert-base-uncased model from the Hugging Face’s transformers library (Wolf et al., 2020).

In the experiments of Section 4.3, we conduct an ablation study that reveals the performance impact of adding/removing some of these features from our models.

#### 3.2 Models

We explore 5 different machine learning models in our experiments:

- Ridge Regression (Ridge) (Tikhonov, 1943)
- Support Vector Machines (SVM) (Boser et al., 1992)
- AdaBoost Regression (AdaBoost) (Freund and Schapire, 1997)
- Gradient Boosting (GBoost) (Friedman, 2001)
- Random Forests (Forests) (Ho, 1995)

The final configuration we chose to submit to LCP 2021 is described in Section 5. In the following section, we explain how we got to that configuration.

### 4 Preliminary Experiments

In this section, we describe the preliminary experiments we conducted in an effort to engineer our final systems for the LCP 2021 shared task. In these experiments, all machine learning models were trained and optimized on the training set and tested on the trial set provided by the organizers. All models were implemented using the Scikit-Learn library (Pedregosa et al., 2011) and optimized using grid search and 5-fold cross validation.

Split	Size	(0, 0)	(0, 1)	(1, 0)	(1, 1)	(0, 2)	(2, 0)	(1, 2)	(2, 1)	(2, 2)
Chi-M	1.5M	0.569	0.302	0.288	0.229	0.254	0.211	0.205	0.183	0.171
Chi-S	1.5M	0.547	0.297	0.287	0.225	0.246	0.215	0.200	0.183	0.170
Chi-MS	3M	0.578	0.316	0.312	0.245	0.261	0.233	0.217	0.200	0.184
Fam-M	2.9M	<b>0.609</b>	0.334	0.318	0.255	0.284	0.232	0.232	0.201	0.191
Fam-S	3.1M	0.578	0.338	0.305	0.253	0.277	0.226	0.225	0.202	0.187
Fam-MS	6M	0.607	0.346	<b>0.327</b>	0.263	<b>0.291</b>	0.241	<b>0.239</b>	0.211	0.198
Com-M	19.3M	0.591	0.333	0.323	0.258	0.280	0.241	0.233	0.210	0.196
Com-S	15.7M	0.578	0.351	0.319	0.268	0.279	0.234	0.229	0.211	0.189
Com-MS	35M	0.571	0.339	0.323	0.264	0.277	0.243	0.233	0.216	0.197
Movies	21M	0.592	0.335	0.327	0.262	0.281	0.244	0.236	0.213	0.198
Series	17M	0.576	<b>0.352</b>	0.321	<b>0.269</b>	0.281	0.238	0.233	0.214	0.193
All	38M	0.570	0.341	0.326	0.267	0.279	<b>0.247</b>	0.236	<b>0.219</b>	<b>0.201</b>

Table 1: Trial set Pearson correlations between complexity scores and frequencies for all SubIMDB splits and n-gram configurations on the single words sub-track.

#### 4.1 Corpora Analysis

Arguably the most important features we use are frequencies. These must be calculated based on a language model trained on a specific corpus, so, as a first step in our engineering process, we decided to conduct an experiment to choose a corpus for the shared task in question.

As evidenced and discussed by Brysbaert et al. (2012) and Paetzold and Specia (2016b), frequencies extracted from spoken text corpora tend to correlate better with word complexity, so we decided to choose the SubIMDB corpora (Paetzold and Specia, 2016b) for our experiment. SubIMDB is a structured corpus extracted from 38,102 subtitles of children, family and comedy movies and series. We created 12 SubIMDB splits for this experiment: Children movies (Chi-M), children series (Chi-S), children movies and series (Chi-MS), family movies (Fam-M), family series (Fam-S), family movies and series (Fam-MS), comedy movies (Com-M), comedy series (Com-S), comedy movies and series (Com-MS), all movies (Movies), all series (Series), and the entire corpus (All). We calculate the Pearson correlation between the trial set complexity scores and n-gram frequencies for all n-gram configurations described in Section 3.1. To do so, we trained 5-gram language models over these splits using KenLM (Heafield, 2011).

The results illustrated in Table 1 are absolute correlation scores for the trial set of the single words sub-track (original values were negative, given that complexity inversely correlates with word frequency). We chose absolute scores to make the table more compact. It can be observed that the (0,

0) configuration (no context) yields the best correlations in every scenario. It can also be noted that, while the family movies split (Fam-M) is best for (0, 0), the remaining configurations tend to benefit from larger splits. Based on that observation, in the experiments that follow, we use family movies to calculate frequencies for single words/phrases and the whole SubIMDB corpus for the remaining n-grams.

#### 4.2 Phrase Compositionality

The next step in our engineering process was to optimize the performance of our submission for the phrases sub-track. For that, we tested the hypothesis that the complexity of a phrase can be more reliably modelled if addressed as a product of the complexity of its words. To do so, we first calculated 3 features from our feature set using 4 different composition functions, then calculated the Pearson correlation between them and the reference complexity scores from the trial set.

The features calculated are: Phrase/word frequency, length, and number of senses. The composition functions are: None (addressing the phrase as a single word), averaging, maximum, and minimum. Frequencies were calculated using a 5-gram language model trained over the entire SubIMDB corpus.

The results in Table 2 show that, overall, employing the principle of compositionality in feature calculation for phrases increases the correlation between classic complexity features and human complexity scores. This is especially true for word senses, given that Wordnet has very few phrases

	None	Avg.	Max.	Min.
Frequency	-0.617	-0.641	<b>-0.650</b>	-0.547
Length	0.482	<b>0.482</b>	0.370	0.461
Senses	-0.125	<b>-0.460</b>	-0.430	-0.420

Table 2: Trial set Pearson correlations for different compositionality settings on the phrases sub-track.

catalogued.

In the subsequent experiments, we employ averaging as the compositionality function in feature calculation for phrases.

### 4.3 Feature Selection

The last step in engineering our submissions was to select a set of features and a machine learning model from the ones described in Section 3. To do so, we conducted a thorough ablation analysis with all models and multiple feature subsets.

Each feature subset is identified by a set of IDs. Each ID describes a feature or group of features. The identifiers are:

- Word/phrase frequency (**F**)
- N-gram frequencies (**N**)
- Word/phrase length (**L**)
- Number of senses (**S**)
- BERT vector (**V**)

The F identifier represents the (0, 0) configuration described in Section 3.1, while the N identifier represents all others. For example, the subset FNLSV contains all features, while the subset FNS does not contain length or the BERT vector.

The results in Table 3 show the results for the feature configurations that we feel were the most relevant for our engineering process. It can be seen that the best performing variant for both single words and phrases is an SVM trained over all features except n-gram frequencies. Models tend to benefit from the inclusion of word length, number of senses, and especially the BERT vector to the feature set. Interestingly, discarding n-gram frequencies tends to improve the models’ performance, especially for single words. This was observed not only in the results of Table 3, but also in many other comparisons we tested, such as FNLSV versus FLSV and FNLS versus FLS.

## 5 Task Results

We based the creation of the final UTFPR systems on the experiments of the previous section. Our final systems are SVMs trained with word/phrase frequencies, word/phrase length, number of senses, and BERT vector (no n-gram frequencies). Compositionality in phrases was handled through averaging. Frequencies were calculated using a 5-gram language model trained over family movies from SubIMDB. Due to a limitation in time availability, the BERT model was used in its original pre-trained form and not optimized for the task at hand.

Table 4 showcases our shared task performance in comparison to the top 3 and bottom 3 systems with respect to Pearson correlation. Our systems for single words and phrases placed 45th out of 55 and 29th out of 38, respectively. Inspecting the instances that featured most discrepancy between gold labels and predictions, we found that our systems had a tendency of both underestimating the complexity of some of the most complex words and phrases (above 0.7 complexity) and overestimating the complexity of the simplest ones (below 0.2). The conservative nature of their predictions seems to be the main reason why our systems did not place higher.

## 6 Conclusions

We presented the UTFPR systems submitted to the Lexical Complexity Prediction shared task of SemEval 2021. Although the placing of our systems were not impressive, we do showcase through our preliminary experiments that employing compositionality can potentially improve the predictions for phrases. We also show that including word length, number of senses, and non-optimized BERT vectors to complexity prediction models can noticeably improve their predictions for both words and phrases. In the future, we intend to test the efficacy of adding BERT vectors optimized for the task at hand to the pool of features of our models.



	FN		F		FL		FLS		FLSV	
	Word	Phrase	Word	Phrase	Word	Phrase	Word	Phrase	Word	Phrase
Ridge	0.605	0.585	0.608	0.577	0.602	0.600	0.622	0.603	0.731	0.580
SVM	0.540	0.542	0.597	0.594	0.623	0.525	0.675	0.568	<b>0.755</b>	<b>0.720</b>
AdaBoost	0.606	0.591	0.603	0.604	0.625	0.591	0.691	0.654	0.710	0.664
GBoost	0.645	0.547	0.597	0.591	0.626	0.532	0.703	0.586	0.732	0.679
Forests	0.579	0.564	0.599	0.469	0.573	0.440	0.684	0.493	0.693	0.673

Table 3: Trial set Pearson correlations for different machine learning models and feature subsets.

System	Words	Phrases
Top 1	0.7886	0.8612
Top 2	0.7882	0.8575
Top 3	0.7790	0.8571
<b>UTFPR</b>	<b>0.6875</b>	<b>0.7601</b>
Bottom 3	0.1807	0.3197
Bottom 2	0.0971	0.2821
Bottom 1	-0.0272	0.1860

Table 4: Pearson correlation obtained by the UTFPR systems on the shared task compared to the top 3 and bottom 3 systems of each sub-task.

## Acknowledgments

We would like to thank the Universidade Tecnológica Federal do Paraná for providing the infrastructure necessary to conduct this research.

## References

- Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.
- Marc Brysbaert, Boris New, and Emmanuel Keuleers. 2012. Adding part-of-speech information to the subtlex-us word frequencies. *Behavior research methods*, 44(4):991–997.
- Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Proceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM; New York.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.
- Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.
- Sian Gooding and Ekaterina Kochmar. 2018. Camb at cwi shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194.
- Nathan Hartmann and Leandro Borges Dos Santos. 2018. Nilc at cwi 2018: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340.
- Kenneth Heafield. 2011. **KenLM: Faster and smaller language model queries**. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland. Association for Computational Linguistics.
- Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.
- Michal Konkol. 2016. Uwb at semeval-2016 task 11: Exploring features for complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1038–1041.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **Roberta: A robustly optimized bert pretraining approach**.
- Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. Ltg at semeval-2016 task 11: Complex word identification with classifier ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000.
- George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to wordnet: An on-line lexical database.

- International journal of lexicography*, 3(4):235–244.
- Gustavo Paetzold and Lucia Specia. 2016a. Anita: An intelligent text adaptation tool. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 79–83.
- Gustavo Paetzold and Lucia Specia. 2016b. Collecting and exploring everyday language for predicting psycholinguistic properties of words. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1669–1679.
- Gustavo Paetzold and Lucia Specia. 2016c. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Gustavo Paetzold and Lucia Specia. 2016d. Sv000gg at semeval-2016 task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974.
- Gustavo Paetzold and Lucia Specia. 2016e. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Andrey Nikolayevich Tikhonov. 1943. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

# RG\_PA at SemEval-2021 Task 1: A Contextual Attention-based Model with RoBERTa for Lexical Complexity Prediction

Gang Rao\*, Maochang Li\*, Xiaolong Hou, Lianxin Jiang, Yang Mo, Jianping Shen

Ping An Life Insurance, Lt

ShenZhen, China

{raogang532, limaochang389, houxiaolong430, jianglianxin769, moyang853, shenjianping324}@pingan.com.cn

## Abstract

In this paper we propose a contextual attention-based model with two-stage fine-tune training using RoBERTa. First, we perform the first-stage fine-tune on corpus with RoBERTa, so that the model can learn some prior domain knowledge. Then we get the contextual embedding of context words based on the token-level embedding with the fine-tuned model. And we use Kfold cross-validation to get K models and ensemble them to get the final result. Finally, we attain the 2nd place in the final evaluation phase of sub-task 2 with Pearson correlation of 0.8575.

## 1 Introduction

LCP is an augmented version of Complex Word Identification(CWI) (Shardlow et al., 2020), predict complexity score for each target word in a sentence. The dataset is a multi-domain English dataset annotated with a 5-point Likert scale (1-5). The annotation model in CompLex addresses complexity as a continuum instead of a binary feature. Previous studies of CWI treat the task as a binary classification, predict a complexity label (complex vs. non-complex) for a set of target words in a sentence.

In this paper, we exploratory data analysis(EDA) for the dataset, and found that the distribution of the single task and multi task dataset are very inconformity, so should bulid two models for every dataset great than one model for merge two task dataset.

Several key technologies as follows:

- Train a RoBERTa based fine-tune corpus classifier. It use the data of all single and multi

---

These authors contributed equally to this work This work is licensed under a Creative Commons Attribution 4.0 International License. License details: <http://creativecommons.org/licenses/by/4.0/>

with train, trial and test dataset as train data, no dev and test, and only train 1 epoch, which enable the RoBERTa model ahead of time learning the domain knowledge.

- At each layer, calculate target vector and context tokens embedding attention, the layer context vector is average the context tokens embedding with soft alignment.
- Weighted the RoBERTa last 12 layers context vector and target vector. They use the same weights, and it's sum equals to 1.
- The degeneration of gradual unfreezing (Howard and Ruder, 2018). At first epoch freeze the pretrained model parameter only learning the head layers parameter, then unfreeze all model parameter.
- Multi-Sample Dropout at last layer (Inoue, 2019)

## 2 Background

Previous approaches to CWI typically refer to binary identification of complex words, two shared tasks on CWI topic have been organized so far. SemEval-2016 Task 11 (Paetzold and Specia, 2016) and BEA workshop 2018 (Yimam et al., 2018). The two tasks approache a number of different model to classification, ranging from traditional machine learning classifiers such as support vector machines (SVM), decision trees, random forest, and maximum entropy classifiers to deep learning classifiers, such as recurrent neural networks. A wide range of features were used such as word embeddings, word and character n-grams, word frequency, Zipfian frequency distribution, word length, morphological, syntactic, semantic, and psycholinguistic.

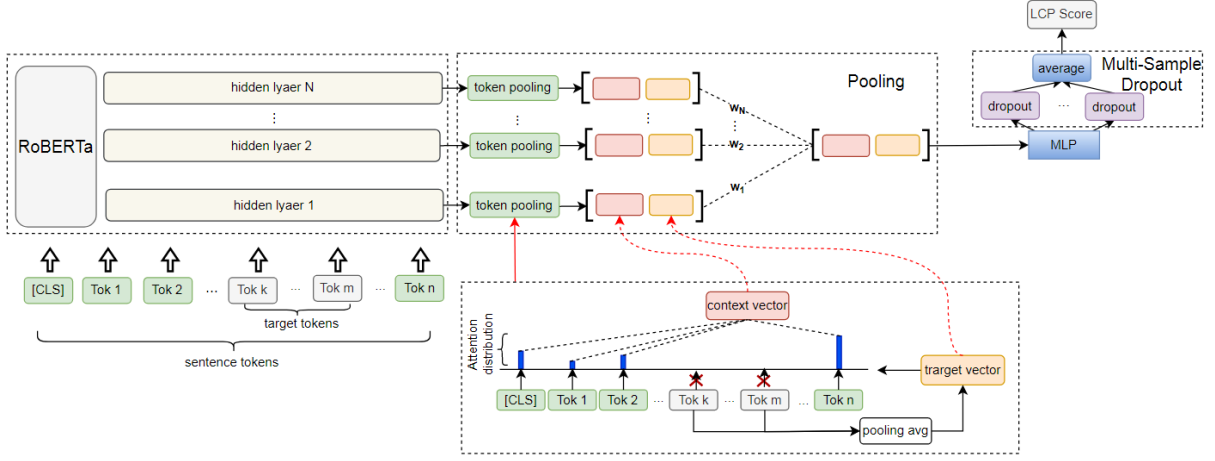


Figure 1: Attention Based Context Representation for LCP

BERT is a new language representation model, and stands for Bidirectional Encoder Representations from Transformers (Devlin et al., 2018). Since BERT appear, fine-tuned pre-trained model with just one additional output layer to create state-of-the-art models for a wide range of tasks. A number of Transformers series models are proposed, such as GPT-2, RoBERTa (Liu et al., 2019), XLM, DistilBert, XLNet. In this papaer we focus on use RoBERTa to slove the tasks.

### 3 System overview

We propose a RoBERTa with attention based model to solve the LCP task, Figure 1 outlines our proposed model framework. First, use Byte-Pair Encoding (BPE) to tokenize the input sentence, which is an effective subword technique to relieve the Out-of-Vocabulary (OOV) problem. For every RoBERTa hidden layer, we apply token pooling that is average the target words tokens embedding as the target vector. Then we calculate the attention between target vector and context tokens vector which are sentence tokens masked the target tokens, After that, context tokens embedding multiply attention weight as the context vector, then concatenate the context vector and target vector. Second, pooling the RoBERTa last 12 layers context vector and target vector, Finally, connect the MLP layer to predict the LCP Score.

#### 3.1 Pooling

The input sentnct is tokenized to  $n$  tokens  $t_i, i = 1, 2, \dots, n$ , and the target tokens index are  $l_t = \{k, k + 1, \dots, m\}$ , the context tokens index are  $l_c = \{1, \dots, k - 1, m + 1, \dots, n\}$ , which exclude

the target tokens.  $E_i^j$  denotes the  $i_{th}$  token embedding of hidden layer  $j$ ,  $T^j$  denotes the target vector of the hidden layer  $j$ .

$$T^j = \frac{1}{m - k + 1} \sum_{i \in l_t} E_i^j \quad (1)$$

The attention weight between context tokens embedding and target vector of layer  $j$  is compute by.

$$\alpha_i^j = softmax(s(E_i^j, T^j)) = \frac{exp(s_i^j)}{\sum_i exp(s_i^j)}, i \in l_c \quad (2)$$

where

$$s_i^j = s(E_i^j, T^j) = (E_i^j)^T T^j \quad (3)$$

After that, we compute the weighted summation for  $c^j$

$$c^j = \sum_{i \in l_c} \alpha_i^j E_i^j \quad (4)$$

Finally, calculate the pooling target vetor  $\bar{t}$ , and context vector  $\bar{c}$ , they are weighted last  $N$  layers target vector and context vector of RoBERTa, the weight  $w_1, w_2, \dots, w_N$  is the model parameters and equals to 1.

$$\bar{t} = \sum_j^N w_j t^j \quad (5)$$

$$\bar{c} = \sum_j^N w_j c^j \quad (6)$$

$$\sum_{j=1}^N w_j = 1 \quad (7)$$

Quantile	Single		Multi	
	train	trial	train	trial
Q1	0.2115	0.2143	0.3026	0.3090
Q2	0.2794	0.2667	0.4091	0.4219
Q3	0.3750	0.3594	0.5294	0.5140

Table 1: Single and Multi DataSet Quantile

## 4 Experiments

### 4.1 Data

Table 1 shows the single and multi dataset quantile. From the table we found that the single and multi words Complexit Score distribution are very different. Single is easy to understand but multi words are difficult. So we build different model to adjust the dataset.

We use 5-fold cross validation, first generate a new feature score\_bin which is binning the LCP score by quantile. Because the dev dataset commonly used to search the optimal hyper parameters, in this experiment we only use dev dataset to found the best epoch, in order to prevent overfitting by early stop, but we found that pretrained model only train 5-6 epochs could be convergence on the task dataset, so not need deliberately generate the dev dataset, only let dev dataset as same as test dataset. Train and test dataset are splited use stratified KFold by the features domain corpus and score\_bin.

### 4.2 Corpus information

The dataset give the sentences domain corpus, but how to use this information? At first, we build the multi-task learning. The auxiliary task is the corpus classification which use the last 12 layers average CLS token embedding. But the auxiliary task not improve the LCP task, and the accuracy of corpus classifier is quite low. It's not conform to the actual, because of the sentences corpus come from Bible Europarl and Biomedical, and they are very easy to distinguish.

Since that, we build a corpus classification model separately which is a RoBERTa fine-tune model (Sun et al., 2019). Benefit from the dataset are easy to classify, the model only need to train 1 epoch, and could get 0.99 accuracy. We merge the single and multi trial, train and test dataset as new train dataset, this can let the model see all data include test dataset. After train, the RoBERTa learning the domain knowledge, and in advance learning part of the test dataset.

Then, export the RoBERTa model as the pre-trained model of LCP task.

### 4.3 Single LCP Task

First merge the single train and trial dataset, then process stratified 5-fold, compare the origin pre-trained model(RoBERTa-large) and fine-tune by the corpus classification(pre-RoBERTa-large).

For train, we use the Mean squared error(MSE) loss function and adam optimizer (Kingma and Ba, 2014). At first epoch we freeze the RoBERTa parameters, only training the head layers. Apply learning rate linear schedule with warmup,  $lr = 2e - 5$ ,  $warmup\_steps = 200$ , and use early stop.

Table 2 shows the single task result, the metric is Pearson correlation (R). The **fold-x** column is the metric of CV model evaluate on the fold-x dataset. The **mean** column is the average of the fold-\* column. Pre-trained corpus classification with fine-tune RoBERTa-large is a little outperformance than origin RoBERTa-large. The single model result is the average of all 5-fold models's predict result for single task test data, and **model result** column is the metric of the the model result. The task final result is the average of all models result, and **final result** column is the metric of the the final result. The two model can achieve 0.7586 and 0.7618, but use simple average ensemble could get 0.7629. It's quite effective.

### 4.4 Multi LCP Task

The merged dataset of multi train and trial only have 1616 examples, In single task, the pre-RoBERTa-large is outperformance than origin RoBERTa-large. In order to augment the multi task examples, Fisrt use the data which merge all sigle and multi train trial dataset, use 5-fold cross validation, splited data use stratified KFold as same as single task. Then use pre-RoBERTa-large train the LCP task. After that, inference the vector  $\bar{h} = [\bar{c}, \bar{t}]$  for all merge data, the final  $\bar{h}$  is average of all 5-fold models. Finally, use the vector to calculate cosine similarity of the multi dataset with single dataset, then recall single examples add to the multi train example with threshold. Here we use  $sim\_threshold = 0.75$ , and recall 2707 single examples.

Then split dataset and train strategy are as same as singe task. The results are in Table 3. gen-RoBERTa-large is the origin RoBERTa model with Data Augmentation, pre-gen-RoBERTa-large is the RoBERTa model fine-tune by the corpus with Data

model	the result of 5-fold cross validateion					LCP single task		
	fold-1	fold-2	fold-3	fold-4	fold-5	mean	model result	final result
RoBERTa-large	0.7528	0.7723	0.7777	0.7854	0.7696	0.7716	0.7586	
pre-RoBERTa-large	0.7636	0.7725	0.7707	0.7849	0.7719	0.7727	0.7618	<b>0.7629</b>

Table 2: Single Task Result

model	the result of 5-fold cross validateion					LCP multi task		
	fold-1	fold-2	fold-3	fold-4	fold-5	mean	model result	final result
RoBERTa-large	0.7531	0.7681	0.7829	0.7692	0.7492	0.7645	0.8310	
pre-RoBERTa-large	<b>0.7862</b>	<b>0.7952</b>	0.7624	0.7352	0.7656	0.7689	0.8332	
gen-RoBERTa-large	0.7713	0.7517	<b>0.7845</b>	0.7550	<b>0.7793</b>	0.7684	0.8325	<b>0.8575</b>
pre-gen-RoBERTa-large	0.7614	0.7646	0.7616	<b>0.7920</b>	0.7791	<b>0.7717</b>	0.8355	

Table 3: Multi Task Result

Augmentation. Results shows model fine-tune by the corpus classification are outperformance than origin model, The final result 0.8575 is fusioned by average of the four models cv results, and rank the 2nd in test phrase.

## 5 Conclusion

This paper presents a method to predicting lexical complexity, which apply RoBERTa-large as the backbone language model. First fine-tune backbone model for corpus classification. Then build model with attention based context representation. make vector recall for multi task data augmentation. Finally, we carry out a multi-model average ensemble strategy to enhance the model performance. In the future, we will exploit better model for text representation, and utilizing data augmentation for all task.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.
- Hiroshi Inoue. 2019. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Gustavo Paetzold and Lucia Specia. 2016. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. Complex: A new corpus for lexical complexity prediction from likert scale data. *arXiv preprint arXiv:2003.07008*.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Computational Linguistics*, pages 194–206. Springer.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of BEA*.

# CSECU-DSG at SemEval-2021 Task 1: Fusion of Transformer Models for Lexical Complexity Prediction

Abdul Aziz, MD. Akram Hossain, and Abu Nowshed Chy

Department of Computer Science and Engineering

University of Chittagong, Chattogram-4331, Bangladesh

{aziz.abdul.cu, akram.hossain.cse.cu}@gmail.com,  
and nowshed@cu.ac.bd

## Abstract

Lexical complexity prediction (LCP) conveys the anticipation of the complexity level of a token or a set of tokens in a sentence. It plays a vital role in the improvement of various NLP tasks including lexical simplification, translations, and text generation. However, multiple-meaning of a word in multiple circumstances, grammatical complex structure, and the mutual dependency of words in a sentence make it difficult to estimate the lexical complexity. To address these challenges, SemEval-2021 Task 1 introduced a shared task focusing on LCP and this paper presents our participation in this task. We proposed a transformer-based approach with sentence pair regression. We employed two fine-tuned transformer models including BERT and RoBERTa to train our model and fuse their predicted score to the complexity estimation. Experimental results demonstrate that our proposed method achieved competitive performance compared to the participants' systems.

## 1 Introduction

Lexical complexity prediction (LCP) has become an important task in this globalization age, especially for second language learners (Przybyła and Shardlow, 2020). LCP is a little bit expansion of complex word identification (CWI) task (Paetzold and Specia, 2016; Štajner et al., 2018), where CWI is a binary classification of a word that is complex or not and LCP is finding the complexity level of a word in continuous labelling in a sentence (Shardlow et al., 2020). LCP plays a vital role in many NLP applications such as lexical simplification (Paetzold, 2016; Paetzold and Specia, 2017; Qiang et al., 2020), text generation, and machine translation (Wang et al., 2016). Besides, it helps those people who are suffering from

Dyslexia (Rello et al., 2013a), Aphasia (Rello et al., 2013b), and those with low literacy levels (Aluisio and Gasperin, 2010).

LCP is a very challenging task (Zampieri et al., 2017), especially because the non-identical target audiences will have distinct needs. For example, speakers of one language usually less familiar with different subsets of the vocabulary of a second language. Besides, the grammatical shape of a sentence and the ambiguous meaning of a word in different places make this task more challenging and important to explore. A single word may portray different lexical complexity because of its non-identical usage, position, tense form, and redundancy in different sentences or in the same sentence. To estimate multi-word complexity, we need to consider the dependency between tokens.

Sentence	Token	Complexity
Sub-task 1		
His head is like the purest gold.	gold	0.210
Sub-task 2		
They shall eat it with bitter herbs.	bitter herbs	0.25

Table 1: Example of sub-task 1 and sub-task 2.

To address the challenges of lexical complexity prediction of words in sentences, (Shardlow et al., 2021a) proposed a shared task at SemEval-2021 Task 1. The task is divided into two subtasks. In sub-task 1, a system needs to determine the complexity level of a word in the sentence, whereas in sub-task 2, a system needs to determine the overall complexity level of multiple words in the sentence. To explain the definition of both sub-tasks, we articulate a few examples in Table 1.

\*\*The first two authors have equal contributions.

Taking part in the LCP shared task of SemEval-2021, we exploit the pairwise contextual information of sentence and token. In this regard, we proposed a combined transformer based framework with sentence pair regression. We make a pairwise learning framework with the sentence-token pair to train the two state-of-the-art transformers model including BERT and RoBERTa.

We organize the rest of the paper as follows: Section 2 presents the details of our proposed framework. Whereas in Section 3, we present our experimental settings and analyze the performance of our model against the various settings and related methods. Finally, we conclude our paper in Section 4 with some future directions.

## 2 Proposed Lexical Complexity Prediction Framework

In this section, we describe our proposed lexical complexity prediction framework. Our goal is to predict the complexity score of a token or a set of tokens in the given sentence. We depict the overview of our framework in Figure 1.

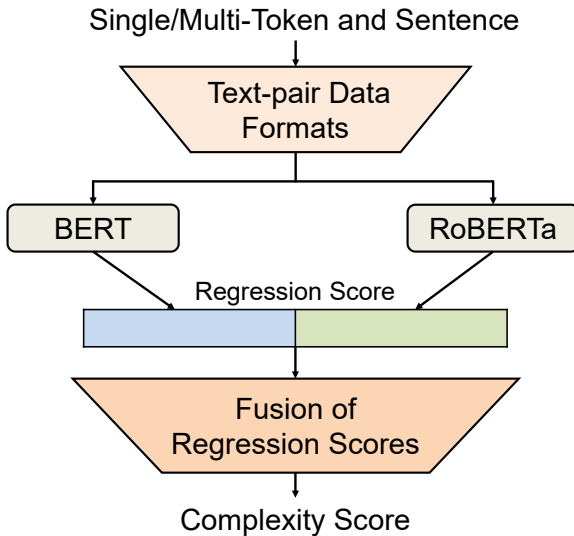


Figure 1: Overview of our proposed framework.

In our framework, we use a sentence pair regression concept in transformer models to perform lexical complexity prediction where input sentence and target word pairs are packed together into a single sequence. After performing sentence-token pair regression through BERT and RoBERTa models, we estimate each model’s regression score. Subsequently, we fuse these models’ predictions by taking the mean of these scores to determine the final complexity score.

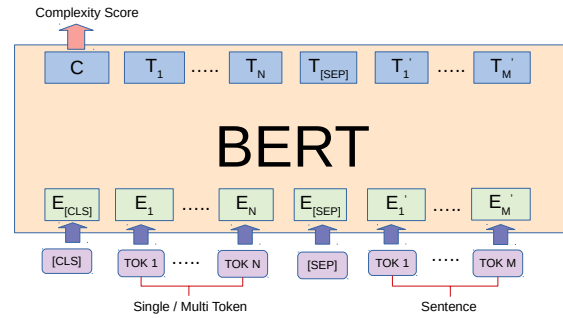


Figure 2: Pairwise learning using BERT model.

### 2.1 Fine-tuned Transformer Models

We fine-tune the transformer models to perform sentence pair regression for LCP through BERT and RoBERTa. We describe the details in the subsequent sections.

#### 2.1.1 Input Representation

We train with the sentence-word pair for better understanding their contextual relation which in turn helps to estimate the complexity of the target word in the sentence. It is important for an LCP system to predict both single and multi-words complexity. We exploit Huggingface transformers library (Wolf et al., 2020) with pairwise training where input target words and sentence make pair as a single sequence and detached with the [SEP] token. We utilize two pre-trained transformer models including RoBERTa and BERT. For LCP tasks training, each model’s first token is the special [CLS] token at the beginning of every sequence which is also responsible for the final layer regression score of each model. For each sequence, we separate every pair with [SEP] token (as presented in Figure 2) where the target words belong to text.a and sentence belongs to text.b. We fine-tune the architecture with the pre-trained BERT and RoBERTa models to estimate the complexity score.

#### 2.1.2 BERT

BERT (Devlin et al., 2019) stands for bidirectional encoder representations from transformers, is a new method of pre-training sentence representations which achieves state-of-the-art results on many NLP tasks including question-answering, text classification, and sentence-pair regression. We take advantage of the bert fast tokenizer and bert-base-uncased model for sentence-pair regression where target words and sentence make pair as a single sequence.



### 2.1.3 RoBERTa

RoBERTa (Liu et al., 2019) is an extension to the original BERT model which is named as a robustly optimized BERT pre-training approach. It focuses on the key hyper-parameters choices and removing the next sentence prediction (NSP) objective. Besides, it is training with much larger mini-batches and learning rates. We exploit the roberta fast tokenizer and roberta-base model for sentence-pair regression to get the complexity score where target word and sentence are trained as pairwise training.

## 2.2 Fusion of Transformer Models

To ameliorate the performance of individual models, we fuse the predicted complexity score of two models to generate a unified score. We use the arithmetic mean to average both model’s regression scores to determine the final complexity score. The estimation is computed as follows:

$$C_i = \frac{x_i + y_i}{2}$$

where  $x_i$  and  $y_i$  correspond to the BERT and RoBERTa regression score, respectively.

## 3 Experiment and Evaluation

### 3.1 Dataset Description

The organizers of the lexical complexity prediction (LCP) task 1 at SemEval-2021 (Shardlow et al., 2021a) provided a multi-domain English benchmark dataset (Shardlow et al., 2020, 2021b) to evaluate the performance of the participants’ systems. The dataset was collected from three different corpora including the Bible, europarl, and biomedical. The proposed task is divided into two subtasks, sub-task 1 focused on single word instances whereas sub-task 2 focused on multi-word instances. The training set for sub-task 1 contains 7662 instances where 2574 instances from Bible, 2576 instances from biomed, and 2512 instances from europarl. The training set of sub-task 2 comprises 1517 instances (505 Bible, 514 biomed, and 498 europarl). The validation set consists of 99 multi-word expressions (29 Bible, 33 biomed and 37 europarl) and 421 single word instances (143 Bible, 135 biomed and 143 europarl). The organizer provided 917 single word instances (283 Bible, 289 biomed, and 345 europarl) for sub-task 1 and 184 multi-word instances (66 Bible, 53 biomed, and 65 europarl) for sub-task 2 as a test set.

### 3.2 Experimental Settings

We now describe the set of parameters that we have used to design our proposed lexical complexity prediction model. In our CSECU-DSG system, we utilize two state-of-the-art Huggingface transformer models with fine-tuning, including BERT and RoBERTa. We use simpletransformers API (Rajapakse, 2019) to implement our system. We train our system with the provided training data. We trained BERT and RoBERTa model using 5 epochs and set the learning rate of 2.99e-5, save\_steps = 767, and evaluate\_during\_training\_steps = 40. We used the CUDA-enabled GPU and set the manual\_seed = 4 to generate the reproducible results. Default settings were used for the other parameters.

### 3.3 Evaluation Measures

To evaluate the performance of participants’ lexical complexity prediction systems, SemEval-2021 task 1 organizers used different strategies and metrics for sub-task 1 and sub-task 2 (Shardlow et al., 2020). For both sub-task, standard evaluation metrics including Pearson correlation (R), Spearman correlation (Rho), mean absolute error (MAE), mean squared error (MSE), and R-squared (R<sup>2</sup>) were applied to estimate the performance of a system. However, Pearson correlation (R) is considered as the primary evaluation measure for both subtasks of this task.

### 3.4 Results and Analysis

The comparative results of our proposed CSECU-DSG system along with top-5 performing systems (Shardlow et al., 2021a) in sub-task 1 and sub-task 2 are presented in Table 2 and Table 3, respectively. Following the benchmark of SemEval-2021 task 1, participants’ systems are ranked based on the primary evaluation metric Pearson correlation (R) score.

At first, we presented the performance of our proposed method. We also presented the performance of top-5 ranked participating systems and LCP baselines. Here, we see that our proposed method obtained competitive performance against the other top-performing systems. In comparison to the other participants’ methods, we have seen that our system demonstrated a similar kind of performance on both sub-task. This deduces the applicability and generalizability of our system for the complexity estimation of both the single and multi-words.

Team (Rank)	Pearson	Spearman	MAE	MSE	R <sup>2</sup>
CSECU-DSG (9th)	0.7716	0.7326	0.0632	0.0066	0.5909
Top performing team based on Pearson correlation score					
JUST BLUE (1st)	0.7886	0.7369	0.0609	0.0062	0.6172
DeepBlueAI (2nd)	0.7882	0.7425	0.0610	0.0061	0.6210
Alejandro Mosquera (3rd)	0.7790	0.7355	0.0619	0.0064	0.6062
Andi (4th)	0.7782	0.7287	0.0637	0.0064	0.6036
CS-UM6P (5th)	0.7779	0.7366	0.0803	0.0100	0.3813

Table 2: Comparative results with other selected participants (Sub-task 1).

Team (Rank)	Pearson	Spearman	MAE	MSE	R <sup>2</sup>
CSECU-DSG (12th)	0.8311	0.8153	0.0678	0.0077	0.6825
Top performing team based on Pearson correlation score					
DeepBlueAI (1st)	0.8612	0.8526	0.0616	0.0063	0.7389
rg_pa (2nd)	0.8575	0.8529	0.0672	0.0072	0.7035
xiang_wen_tian (3rd)	0.8571	0.8548	0.0675	0.0072	0.7012
andi_gpu (4th)	0.8543	0.8448	0.0664	0.0071	0.7055
ren_wo_xing (5th)	0.8541	0.8473	0.0677	0.0073	0.6967

Table 3: Comparative results with other selected participants (Sub-task 2).

### 3.5 Discussion

In order to estimate the effect of each component of our CSECU-DSG model, we estimated the performance of the individual model. The summarized experimental results for sub-task 1 and sub-task 2 are presented in Table 4.

From the results, it can be observed that RoBERTa based model performed better compared to the BERT model when considering individual model’s performances. However, combining two models regression scores by using mean increased Pearson correlation score by more than 1% on both subtasks. This deduced the importance of our model fusion.

All three models performed better for multi-words complexity estimation compared to the single word complexity. We have seen a similar kind of trend in other models’ performances reported in Table 2, and Table 3. This demonstrated that estimating the single word complexity is more challenging compared to the multi-words expression. This is because a multi-word expression contains more words, therefore, contains more contextual information that helps the model for complexity estimation compared to the single word.

Method	Single Word	Multi Word
	Pearson	Pearson
CSECU-DSG	0.7716	0.8311
Performance of Individual Model		
–BERT	0.7514	0.8077
–RoBERTa	0.7634	0.8211

Table 4: Performance analysis of individual model.

## 4 Conclusion and Future Directions

In this paper, we presented our approach to the lexical complexity prediction task. We tackled the problem by performing sentence pair regression using two SOTA transformer models including BERT and RoBERTa in a unified architecture. By using pairwise learning, we exploited the contextual relation between sentence-word pairs to estimate the complexity score. Our method achieved competitive scores compared to other participants.

In the future, we have a plan to incorporate various handcrafted features with state-of-the-art neural methods to distill the relationship of sentence-word pairs for complexity estimation.

## References

- Sandra Aluisio and Caroline Gasperin. 2010. Fostering digital inclusion and accessibility: the PorSimples project for simplification of Portuguese texts. In *Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas*, pages 46–53.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 4171–4186.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, pages arXiv–1907.
- Gustavo Paetzold and Lucia Specia. 2016. SemEval 2016 Task 11: Complex Word Identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Gustavo H Paetzold and Lucia Specia. 2017. A survey on lexical simplification. *Journal of Artificial Intelligence Research*, 60:549–593.
- Gustavo Henrique Paetzold. 2016. *Lexical Simplification for Non-Native English Speakers*. Ph.D. thesis, University of Sheffield.
- Piotr Przybyła and Matthew Shardlow. 2020. Multi-Word Lexical Simplification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1435–1446.
- Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2020. Lexical simplification with pretrained encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34,05, pages 8649–8656.
- T. C. Rajapakse. 2019. Simple Transformers. <https://github.com/ThilinaRajapakse/simpletransformers>.
- Luz Rello, Ricardo Baeza-Yates, Laura Dempere-Marco, and Horacio Saggion. 2013a. Frequent words improve readability and short words improve understandability for people with dyslexia. In *IFIP Conference on Human-Computer Interaction*, pages 203–219. Springer.
- Luz Rello, Susana Bautista, Ricardo Baeza-Yates, Pablo Gervás, Raquel Hervás, and Horacio Saggion. 2013b. One half or 50%? An eye-tracking study of number representation readability. In *IFIP Conference on Human-Computer Interaction*, pages 229–245. Springer.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A New Corpus for Lexical Complexity Prediction from Likert Scale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting Lexical Complexity in English Texts. *arXiv preprint arXiv:2102.08773*.
- Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.
- Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30,1.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in Data Annotation and System Performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*.

# CLULEX at SemEval-2021 Task 1: A Simple System Goes a Long Way

Greta Smolenska, Peter Kolb, Sinan Tang,  
Mironas Bitinis, Héctor Hernández, Elin Asklov

Babbel|Lesson Nine GmbH

{gsmolenska, pkolb, stang, mbitinis, hhernandez, easkloev}@babbel.com

## Abstract

This paper presents the system we submitted to the first Lexical Complexity Prediction (LCP) Shared Task 2021. The Shared Task provides participants with a new English dataset that includes context of the target word. We participate in the single-word complexity prediction sub-task and focus on feature engineering. Our best system is trained on linguistic features and word embeddings (Pearson's score of 0.7942). We demonstrate, however, that a simpler feature set achieves comparable results and submit a model trained on 36 linguistic features (Pearson's score of 0.7925).

## 1 Introduction

Lexical complexity relates to complexity of words. Its assessment can be beneficial in a number of fields, ranging from education to communication. For instance, lexical complexity studies can assist in providing language learners with learning materials suitable for their proficiency level or aid in text simplification (Siddharthan, 2014). These studies are also a central part of reading comprehension, as lexical complexity can predict which words might be difficult to understand and could hinder the readability of the text. Lexical complexity studies typically make use of Natural Language Processing and Machine Learning methods (Paetzold and Specia, 2016).

Previous similar studies focus on Complex Word Identification (CWI), which is a process of identifying complex words in a text (Shardlow, 2013). In this case, lexical complexity is assumed to be binary - words are either complex or not. LCP Shared Task 2021 addresses this limitation by introducing a new dataset designed for *continuous* rather than *binary* complexity prediction (Shardlow et al., 2021).

In this paper, we describe a *single-word* lexical complexity prediction system. Our goal is to

demonstrate that a simple system can achieve results comparable to more complex ones. Therefore, we focus on feature engineering rather than model tuning.

## 2 Related Work

### 2.1 Lexical Complexity

Over the years, studies on lexical complexity have ranged from research on the overall *readability* enhancement and *text simplification* to studies focusing specifically on lexical complexity.

Some of the earlier work on lexical complexity targeted communication enhancement of medical documents by assessing the familiarity of medical terminology (Zeng et al., 2005). Paetzold and Specia (2013) showed that the absence of lexical simplification in Automatic Text Simplification (ATS) systems yielded texts that readers might still find too complex to understand.

CWI has then gained more interest, and two Shared Tasks have been organised with the goal of establishing state-of-the-art performance in the field. SemEval-2016 Task 11 approached CWI as a *binary* classification task and collected a dataset for English which was annotated by non-native speakers (Paetzold and Specia, 2016). Zampieri et al. (2017) showed that such data annotation approach was not optimal. The second Shared Task addressed the limitations by introducing a multi-lingual dataset for Spanish, German, English and French and approaching the problem as both, a *binary* and a *probabilistic* complexity prediction task (Štajner et al., 2018).

### 2.2 Feature and Model Selection

In lexical complexity prediction tasks, linguistic features and word frequency measures have been proven to be among the most effective features. The winning systems developed for the CWI 2018

Shared Task (Yimam et al., 2018) use various lexical features, such as word N-gram, POS tags, and syntactic dependency parse relations. Moreover, they also include different variants of word frequency features, CEFR levels, and a few more.

As for the choice of algorithms, Gooding and Kochmar (2018) has achieved the best performing systems in English monolingual tasks using classifiers with ensemble techniques, such as AdaBoost with 5000 estimators and the aggregation classifier of Random Forest. The winning systems for multilingual tracks (Kajiwara and Kochi, 2018) also employ random forest models.

### 3 LCP Shared Task 2021 Setup

The LCP Shared Task 2021 aims to predict the complexity value of words in their context. It is divided into two sub-tasks: predicting the complexity score of 1) *single words* and 2) *multi-word expressions*. In this paper, we present a system for the first sub-task.

The Shared Task uses the CompLex corpus (Shardlow et al., 2020). In addition to the target word, it includes contextual information which is represented by a sentence where the word appears and its source or domain: *Bible* (Christodouloupoulos and Steedman, 2015), *Europarl* (Koehn, 2005) or *biomedical texts* (Bada et al., 2012). Each word in the dataset is evaluated by around 7 annotators from English speaking countries. The complexity labels are based on a 5-point Likert scale scheme (*very easy* to *very difficult*). The final dataset consists of 7,662 training and 917 testing instances.

The Shared Task baseline system uses a linear regression model. It is trained on log relative frequency and word length features, resulting in a Mean Absolute Error (MAE) of **0.0867**.

## 4 Methodology

In this section, we describe the methodology that we follow in the design of our system, including the used data, feature engineering and the training steps. The study relies on an in-depth experimentation with features. We aim to find out which linguistic information is the best predictor of lexical complexity.

### 4.1 Data Collection

For the computation of some features, we use additional data sources. We extract word frequencies from nine corpora that cover different do-

main and complexity levels: *BNC corpus*<sup>1</sup>, *Simple Wikipedia and English Wikipedia*<sup>2</sup>, *SubIMDB*<sup>3</sup> and English monolingual corpora from the *OPUS project*<sup>4</sup>: *bible-uedin*<sup>5</sup>, *EMEA*<sup>6</sup>, *Europarl*<sup>7</sup>, *News-Commentary*<sup>8</sup> and *OpenSubtitles 2018*<sup>9</sup>. We additionally use two word lists with annotated CEFR levels (Common European Framework of Reference for Languages, which organises language proficiency in six levels, A1 to C2)<sup>10</sup> and the *Age of Acquisition* dataset<sup>11</sup>.

### 4.2 Features

We consider a) word and sentence-level features (or *linguistic* features), b) *frequency* features and c) *word embeddings*.

On a word level, we compute the linguistic information, i.e. *character*, *syllable* and *phoneme counts*, *universal part-of-speech* tag and *named entity* tag (extracted with Stanza NLP toolkit) (Qi et al., 2020). We also compute scores that pertain to language learning such as *age of acquisition*, *percentage of population that knows the word* and *word prevalence* (Kuperman et al., 2012). Finally, we use two CEFR word lists and split them into five subsets each (one per CEFR level). Each word is assigned a *boolean value* depending whether it appears in one of the subsets.

On a sentence level, lexical complexity is represented by *lexical diversity rate* (*unique* words divided by *all* words). Syntactic complexity and readability are represented by the *average sentence length* and the *Linsear Write* score, which is a readability measure used to assess the difficulty of U.S. military manuals (Klare, 1974). We also make special use of the *OpenSubtitles* frequencies: *vocabulary percentage per CEFR level* is computed by splitting the corpus into five subsets and represents the distribution of words among the five frequency ranges; *difficult word percentage* relates to words containing two and more syllables that do not appear in top 200 most common words in the corpus; *unknown word percentage* represents

<sup>1</sup>BNC

<sup>2</sup>Wikipedia Monolingual Corpora

<sup>3</sup>SubIMDB

<sup>4</sup>OPUS resources

<sup>5</sup>bible-uedin

<sup>6</sup>EMEA

<sup>7</sup>Europarl

<sup>8</sup>News-Commentary

<sup>9</sup>OpenSubtitles 2018

<sup>10</sup>The Oxford 5000 and Kelly list for English

<sup>11</sup>AoA

the percentage of words that do not appear in the corpus at all. The final *text complexity score* is a normalised sum of all sentence-level scores.

Additionally, we calculate different types of *frequencies*, i.e. log relative, absolute (raw), frequency rank (word rank in a frequency list) and ZIPF frequency (Zipf, 1949), from the nine corpora.

Finally, we experiment with pre-trained *word embeddings*, including fastText for English and BERT’s embeddings (Mikolov et al., 2018; Devlin et al., 2018). However, we ablate fastText word embeddings from the final feature set as they slightly degrade the overall performance.

### 4.3 Training, Tuning & Testing

The focus of our study is to achieve the best results through feature engineering rather than model hyperparameter tuning. During all experiments, we utilise the open source Machine Learning software WEKA (Frank et al., 2016) with the default algorithm hyperparameter settings and apply 10-fold cross-validation.

#### 4.3.1 Models

First, we select several Machine Learning algorithms for further experiments with the features. During this step, we use *word* and *sentence-level* features with a subset of *frequency* features.

Due to the nature of the dataset target values, we employ classifiers suitable for regression tasks. Specifically, we use `linear regression` and `Multi-Layer Perceptron`, meta classifiers, such as `Bagging`, `Stacking` and `Random Subspace`, and decision trees, such as `M5P` and `Random Forest`. We obtain the best result and benchmark our approach with `M5P` - a model tree algorithm used for numeric prediction (Table 1). We reach MAE of **0.0638** (Pearson’s score of 0.7811), outperforming the baseline model of the Shared Task (Section 3).

Next, we experiment with different feature groups and combinations with the goal to select the optimal feature subset. We train with the five best performing algorithms in each step but report only the results of the best model.

#### 4.3.2 Ablation Studies

We narrow down the selection for the best performing features based on the three feature groups: *frequency* features, *linguistic* features and *word embeddings*.

Classifier	Pearson	MAE
M5P	<b>0.7811</b>	<b>0.0638</b>
Random SubSpace	0.77	0.0657
Bagging	0.7693	0.0657
Random Forest	0.7655	0.0661
Decision Table	0.7601	0.0665

Table 1: 10-fold cross-validation results on the training set for the top 5 classifiers

We pay special attention to frequency features since the previous work shows that word frequencies are usually among the most informative features (Yimam et al., 2018). First, to figure out the best way to represent frequencies of lower cased word forms, we train the M5P model on different frequency representations: *log relative*, *raw*, *ZIPF* and *frequency rank*. We use only the best frequency representation, log relative frequency, in the following steps. We then test the models with frequencies from various sources.

We also conduct experiments to understand the impact of word embeddings using 300-dimension pre-trained word vectors<sup>12</sup>, and BERT<sup>13</sup> embeddings, where we concatenate layers 7 and 11 (Chronis and Erk, 2020) which gives better results than concatenating or summing the last four hidden layers.

We then conduct the final ablation study. Given the complete set of features, we employ WEKA’s feature selection algorithms and remove the least informative features, one feature at a time. In case it does not result in an improvement, the feature is added back and we continue with the next available feature.

## 5 Results

In this section, we present our results and discuss the key findings. All discussed systems are trained with the `Random Forest` classifier.

### 5.1 Frequency Features

We find that a combination of frequency features from different sources alone can result in high performance (Table 2). In this case, daily spoken language sources, such as film subtitles, seem to be the most informative. However, adding more frequency features does not necessarily improve the results (Tables 2 and 3).

<sup>12</sup>fastText for English

<sup>13</sup>We use `bert-base-uncased` from Hugging Face (Wolf et al., 2020)

Frequency Sources	Pearson
All - EMEA	<b>0.713</b>
All	0.7128
All - EMEA - Bible	0.7041
OpenSubs + BNC	0.6882
+ EnWiki + SimpleWiki	
OpenSubs	0.6536
SubIMDB	0.6479

Table 2: Frequency Sources

Features	Pearson
9 frequencies + corpus + POS + syllCount + charCount (13 features)	0.764
Above + BERT 7-11 (1550 features)	0.6953
9 frequencies + corpus + POS + sentence features - depRel - distToHead - NER (44 features)	0.7907
Above - imdbFreq	0.7909
Above - CEFR vocabulary percentages	0.7921
Above - freqPm	0.7924
Above - harmonicMeanDiff (36 features)	<b>0.7925</b>
Above + best BERT 7-11 (76 features)	<b>0.7942</b>

Table 3: Feature Ablation Experiments

## 5.2 Linguistic Features

During the experiments with the linguistic features, we obtain the best results using a reduced 36 feature combination (Table 4). We find out that syntactic features such as target word *distance* to the syntactic head of the sentence and its *syntactic relation* to the head of the sentence seem to worsen the performance (Table 3). The full list of ablation steps can be found in Appendix A.

Furthermore, removing the *sentence-level* features results in a slight decrease of the overall performance (from Pearson’s score of 0.7925 to 0.7791). It indicates that either word-level information remains the most informative for this task or that a single sentence does not provide sufficient contextual information.

## 5.3 Word Embedding Features

Table 4 shows results for the best systems that are trained on *linguistic* features only, *word embedding* features only and the *combined* set of features.

The system trained on the word embeddings performs significantly worse than the other two systems. BERT embeddings only improve the result if we select a subset of 76 out of the 1536 embedding features with WEKA’s `CfsSubsetEval` (Hall, 1999). The model trained on the combined set of features performs the best, reaching Pearson’s score of **0.7942**. However, the difference between this system and the one trained on linguistic fea-

Feature Combination	#Features	Pearson
36 Linguistic + 76 Embedding	112	<b>0.7942</b>
Linguistic	36	0.7925
BERT Embeddings	1536	0.6999

Table 4: Best systems trained on linguistic, word embedding and the combined features

tures is statistically insignificant. These results indicate that word embeddings are less informative than the linguistic information. Additionally, word embedding computation can be costly in terms of the added complexity and the computational resources. We, therefore, argue that a simpler feature combination is sufficient and submit our second best model to the Shared Task.

## 5.4 Test Set

The submitted system that is trained on 36 linguistic features (Appendix B) is evaluated on the official Shared Task test set and reaches Pearson’s score of **0.7588**, ranking in the upper half of the submitted systems.

## 6 Conclusion

In this paper, we have described the design of our system submitted to the LCP Shared Task 2021 and discussed the key findings of our feature engineering approach. We aimed to design a simple system that would not require much classifier tuning or complex feature computations. Our two best models are trained on the Random Forest classifier with the default hyperparameters. The best system is trained on a 112 feature set which includes word embeddings. The second best system is trained on a simple 36 linguistic feature set. We submit the simple system since the performance difference between the two systems is not significant. The model is placed in the upper half of the Shared Task rankings for the single-word prediction subtask (Pearson’s score of 0.7588), demonstrating how a simple approach can achieve high performance results.

Further analysis of the feature ablation studies confirms that word frequencies seem to be the most informative among all features. We also observe that even though including contextual information does improve the overall result, the performance differences are small. Future research might therefore look into including more contextual information than one sentence. In addition, the perception

of word complexity differs from reader to reader. Future work could target specific reader groups, such as people with dyslexia or second language learners. In this case, the relevant background information of the readers should be included in the annotation and experimentation processes.

## References

- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the CRAFT corpus. *BMC bioinformatics*, 13(1):1–20.
- Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: The Bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.
- Gabriella Chronis and Katrin Erk. 2020. When is a bishop not like a rook? When it’s like a rabbi! Multi-prototype BERT embeddings for estimating semantic relationships. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 227–244.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Eibe Frank, Mark A. Hall, and Ian H. Witten. 2016. *The WEKA Workbench*. Morgan Kaufmann. Fourth Edition.
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Mark Andrew Hall. 1999. Correlation-based feature selection for machine learning.
- Tomoyuki Kajiwara and Mamoru Komachi. 2018. [Complex word identification based on frequency in a learner corpus](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 195–199, New Orleans, Louisiana. Association for Computational Linguistics.
- George R Klare. 1974. Assessing readability. *Reading research quarterly*, pages 62–102.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.
- Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 English words. *Behavior research methods*, 44(4):978–990.
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhresch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.
- Gustavo Paetzold and Lucia Specia. 2013. Text simplification as tree transduction. In *Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology*.
- Gustavo Paetzold and Lucia Specia. 2016. SemEval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A Python Natural Language Processing Toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex—A new corpus for lexical complexity prediction from Likert scale data. *arXiv preprint arXiv:2003.07008*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.
- Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.



Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*.

Qing Zeng, Eunjung Kim, Jon Crowell, and Tony Tse. 2005. A text corpora-based estimation of the familiarity of health terminology. In *International Symposium on Biological and Medical Data Analysis*, pages 184–192. Springer.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

## Appendices

### A Feature Ablation Experiments

Features	#Features	Pearson	Features permanently removed
9 frequencies + corpus + POS + syllCount + charCount	13	0.764	
Above + BERT 7-11	1550	0.6953	
9 frequencies + corpus + POS + sentence-level features (- deprel - distToHead - NER)	44	0.7907	yes
above + 300 fastText word embeddings	344	0.766	
44 - imdbFreq	43	0.7909	yes
43 - Oxford lists - Kelly lists	32	0.7902	
43 - AoA	42	0.7891	
43 - CEFR vocabulary percentages	38	0.7921	yes
38 - avgSentenceLength	37	0.792	
38 - linsearWrite	37	0.7917	
38 - unknownWordPercentage	37	0.7906	
38 - difficultWordPercentage	37	0.7914	
38 - lexicalDiversityRate	37	0.792	
38 - textComplexityScore	37	0.7919	
38 - countPhones	37	0.7918	
38 - percKnown	37	0.7908	
38 - freqPm	37	0.7924	yes
37 - prevalence	36	0.79	
37 - freqZipfUS	36	0.7923	
37 - avgDiffRating	36	0.7923	
37 - harmonicMeanDiffRating	36	<b>0.7925</b>	yes
36 + best BERT 7-11 (76)	112	<b>0.7942</b>	yes

## B Final Feature Set

Feature	Description
corpus	One of {bible, biomed, europarl}
POS	Part-of-speech tag
linsearWrite	readability measure used in U.S. military
avgSentenceLength	number of words in the sentence
unknownWordPercentage	unknown word percentage
difficultWordPercentage	difficult word percentage
lexicalDiversityRate	type token ratio (unique words/all words)
textComplexityScore	normalised sum of all sentence-level scores
countPhones	count of phones in word
AoA	age of acquisition
percKnown	Percentage of population that knows the word.
prevalence	word prevalence
freqZipfUS	ZIPF frequency calculated from the AoA dataset
avgDiffRating	Average of difficulty ratings from SVL 12000 dataset
kelly_a1 oxford_a1 kelly_a2 oxford_a2 kelly_b1 oxford_b1 kelly_b2 oxford_b2 kelly_c1 oxford_c1 kelly_c2	<i>boolean: for word that occurs in the CEFR wordlist</i>
syllCount	number of syllables in the word
charCount	number of characters in the word
Europarl_log_rel_freq BNC_log_rel_freq OpenSubs_log_rel_freq SimpleWiki_log_rel_freq EnWiki_log_rel_freq SubIMDB_log_rel_freq News_Comm_log_rel_freq bible_log_rel_freq	<i>log relative frequency of word in the corpus</i>
complexityTargetClass	numeric

# RS\_GV at SemEval-2021 Task 1: Sense Relative Lexical Complexity Prediction

**Regina Stodden**

Dept. of Computational Linguistics  
Heinrich Heine University  
Düsseldorf, Germany  
regina.stodden@hhu.de

**Gayatri Venugopal**

Symbiosis Institute of  
Computer Studies and Research  
Symbiosis International (Deemed University)  
Pune, Maharashtra, India  
gayatri.venugopal@sicsr.ac.in

## Abstract

We present the technical report of the system called RS\_GV at SemEval-2021 Task 1 on complexity prediction of English words. RS\_GV is a neural network using hand-crafted linguistic features in combination with character and word embeddings to predict the target words' complexity. For the generation of the hand-crafted features, we set the target words in relation to their senses. RS\_GV predicts the complexity well of biomedical terms but it has problems with the complexity prediction of very complex and very simple target words.

## 1 Introduction

Text simplification is the process of modifying a text so that it becomes easy for the reader to understand the meaning of the text without any loss of information. A main part of text simplification is lexical simplification. In lexical simplification, complex words are replaced with easier or more frequent synonyms. Following [Shardlow \(2014\)](#), the process of lexical simplification can be split as follows: I.) identification of complex words in a given text, II.) substitution generation, III.) word sense disambiguation, IV.) synonym ranking, V.) substitution of complex word with the best synonym in correct morphological form.

Following [Shardlow \(2014\)](#), the most common errors in lexical simplification are that the words are not identified as complex or that words are incorrectly identified as complex. One reason might be the approach to predict complex words. So far, in the task called *complex word identification* (CWI), a word in a sentence was labeled as either complex or simple without any range in between. [Shardlow et al. \(2020\)](#) criticize this approach because there is no clear threshold when a word starts to be complex. Hence, they propose a new task called *lexical complexity prediction* (LCP). The aim of

LCP is to predict the complexity of a single word or a multi-word expression on a scale of 0 to 1.

This paper proposes RS\_GV, a model for LCP in the context of the SemEval-2021 task 1 ([Shardlow et al., 2021a](#)). RS\_GV uses hand-crafted features relative to their WordNet senses, Flair embeddings and a neural regressor in a cross-domain and within-domain setting.

## 2 Related Work

Lexical complexity prediction is a new sub-task of lexical text simplification. The aim is to predict the complexity of a single word or a multiword expression on a scale of 0 to 1. The most similar task is CWI. In contrast to LCP, CWI aims at binary classification that determines whether a word is complex or not. As LCP has been mentioned for the first time in the context of this shared task ([Shardlow et al., 2020, 2021a,b](#)), no other related work exists yet. Hence, we outline the state of the art in CWI.

**SemEval-2016 Task 11: CWI** [Paetzold and Specia \(2016\)](#) collated 9200 sentences from the CW Corpus ([Shardlow, 2013](#)), the LexMTurk Corpus ([Horn et al., 2014](#)), and the Simple Wikipedia corpus ([Kauchak, 2013](#)). All these corpora were based on the Simple English Wikipedia (SEW). CWI was treated as a binary classification task, wherein 400 non-native speakers annotated content words in English text. It was observed from the annotations that complex words were shorter, less ambiguous and had a low occurrence in SEW. F-score and G-score were used as the evaluation metrics. The features incorporated by the submitted systems can be seen in [Figure 1](#).

It is shown that the word frequency, lexical, semantic and morphological features play a dominant role in CWI. Besides these, n-gram features were also experimented with by a few systems. Word embeddings were not used extensively.

**CWI Shared Task 2018** Another shared task on complex word identification was organized in 2018 (Yimam et al., 2018). Yimam et al. (2018) collected data from three sources, i.e., professionally written news, WikiNews and Wikipedia, and in four languages, i.e., English, German, French, and Spanish. The shared task was composed of two sub-tasks. Sub-task 1 approached the problem as a binary classification problem and sub-task 2 treated it as a probabilistic classification problem, wherein the score between 0 and 1 indicated the proportion of annotators who considered a word as complex. Native as well as non-native readers annotated the dataset created by Yimam et al. (2017). A word was deemed to be complex if at least one out of twenty annotators labeled it as complex. Based on annotations, it was observed that the systems might perform better when trained on domain-specific data. It was also found that traditional feature engineering-based approaches performed better than neural network and word embedding based approaches. The features incorporated by the submitted systems of 11 teams can be seen in Figure 1.

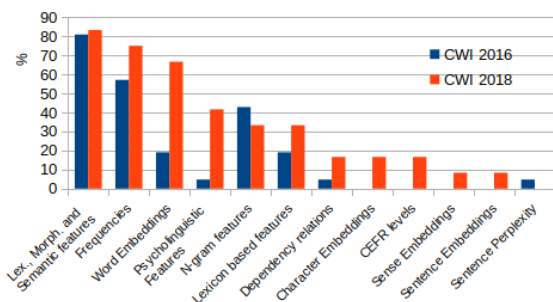


Figure 1: Features incorporated by the systems submitted to CWI Shared Task 2016 (Paetzold and Specia, 2016) and 2018 (Yimam et al., 2018).

The graph reinstates the fact that frequencies, lexical, semantic and morphological features play a key role in CWI. However, it was observed that as compared to 2016, in 2018, word embeddings were more commonly used.

### 3 Experimental Setup

#### 3.1 Data

The corpus (Shardlow et al., 2020, 2021b) contains 9,476 annotated instances in three new CWI/LCP domains, i.e., bible, political and biomedical texts. For every instance, one target word, its target complexity value and its containing sentence are given.

The complexity value is based on crowd-sourced human ratings of at least 4 and at most 20 persons with residence in the UK, USA, or Australia. Each instance was rated on a 5-point Likert scale from 1 (very easy) to 5 (very difficult). Afterwards, the ratings were averaged and normalized on a continuous scale between 0 and 1, where 0 is easy and 1 is complex.

Each target word occurs in multiple instances and may capture different senses so that each word can be assigned to different complexity values in different instances. For example, *vision* occurs in all sub-domains with different meaning, e.g., ability to see, supernatural experience, and foresight.

Following the corpus description (Shardlow et al., 2020), a target word should only occur in a different sentence but not in the same sentence twice. Unfortunately, in our corpus analysis, we found a few doubled instances but with varying complexity values. For example, *body* is rated within in the same sentence in the biomedical part of the set with complexity values of 0.05 and 0.32 (see Appendix C, Table 9). This variation underlines that LCP is a subjective task, and, hence, a difficult NLP task (see section 5.3).

More details regarding the data, including the data split in training, trial, and test can be found in the shared task paper (Shardlow et al., 2021a).

As a preprocessing step we tokenized the sentences and annotated the tokens with their lemma, part-of-speech, and morphological information using spaCy (Honnibal and Montani, 2017). This linguistic information is the basis of our features.

#### 3.2 Evaluation

The lexical complexity prediction is evaluated, following the shared task instructions (Shardlow et al., 2021a), with e.g., Pearson’s correlation ( $r$ , mainly reported here) and Mean Absolute Error ( $MAE$ ).

#### 3.3 Baselines

We use the baseline results reported by the organizers<sup>1</sup> as comparative results. They use linear regression models with the following features, complexity-average, word length, log word frequency from SUBTLEX and log word frequency combined with word length.

<sup>1</sup>[https://competitions.codalab.org/competitions/27420#learn\\_the\\_details-evaluation](https://competitions.codalab.org/competitions/27420#learn_the_details-evaluation)

## 4 System Description

Our system’s main characteristics are a combination of hand-crafted features, contextualized character embeddings (see subsection 4.1), a sense relative normalization (see subsection 4.2), and a neural network for regression (see subsection 4.4).

### 4.1 Features

Based on the survey of features previously used for complexity estimation of words (see section 2), we decided to combine hand-crafted features and contextualized embeddings. A list of all language resources used for feature generation is provided in Appendix B (see Table 7).

#### 4.1.1 Word and Character Embeddings

Similar as proposed in Gooding and Kochmar (2019); Hartmann and dos Santos (2018), and De Hertog and Tack (2018) we use word and character embeddings. We compare pre-trained non-contextualized word embeddings, i.e., GloVe (Pennington et al., 2014), pre-trained contextualized word embeddings, i.e., ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019), with pre-trained contextualized character embeddings, i.e., stacked Flair (Akbik et al., 2018, 2019a) –a combination of GloVe and Flair– and PooledFlair (Akbik et al., 2019b).

We suggest that the contextualized embeddings perform better on LCP as the context of the target word and the meaning of the sentence are important for words’ complexity. To the best of our knowledge contextualized character embeddings have not been used for CWI or LCP before.

The embeddings are extracted using FLAIR (Akbik et al., 2019a). Details regarding the settings of the word and character embeddings are provided in Appendix B (see Table 8).

#### 4.1.2 Hand-crafted Features

An overview of all hand-crafted features used is visualized in Table 1.

**Readability Assessment Features.** We use the sentence’s readability as a feature because we assume that a token would be perceived as more complex if the entire sentence is complex. We implemented the readability using readability scores which are mainly applicable on texts such as Kincaid et al. (1975), Gunning (1952), Coleman and Liau (1975), Dale and Chall (1948) and Senter and Smith (1967) using textstat (Bansal and Aggarwal,

category	feature	category	feature
Readability Assessment	flesch kincaid grade	Morphological	proper noun
	gunning fog		singular
	coleman liau index		plural
	dale chall readability score		famsize
	automated readability index		HAL frequency
	difficult words		number morphemes
Lexical	*frequency	number prefixes	
	*word length	number roots	
	*number consonants	number suffixes	
	*number vowels	suffix length	
	*number syllables	prefix length	
WordNet	*number hypernyms	*familiarity	
	*number hyponyms	*concreteness	
	*number senses	*imagery	
Lexicon	in wordlists	Psycholinguistic	*m.fullness colorado
Other	named entity		*m.fullness pavio
	word position		*age of acquisition

Table 1: List of all used features sorted by category. An asterisk (\*) indicates whether the feature is normalized relative to its senses or not.

2014). We do not consider readability scores that are applicable on sentences as we could not reproduce certain sentence-level readability methods.

**Lexical Features.** Word length, word frequency and number of syllables are included in the set of lexical features following the methodology explained in Shardlow et al. (2020). The word frequency values are obtained from Sharoff (2006) and the GoogleWeb1T resource (Brants, Thorsten and Franz, Alex, 2006). Besides these, the number of consonants and vowels are also calculated.

**WordNet Features.** Paetzold and Specia (2016) use the number of senses, synonyms, hypernyms and hyponyms among other features to identify complex words. In our study, the number of hypernyms, hyponyms and senses are retrieved from the English WordNet (Fellbaum, 1998).

**Psycholinguistic Features.** Similarly as proposed in Davoodi and Kosseim (2016), we generate psycholinguistic features, e.g., word familiarity and age of acquisition, using the Medical Research Council (MRC) Psycholinguistic Database version 2.0 (Wilson, 1988).

**Morphological Features.** As seen in the survey of CWI shared task, morphological features are often used for this task. Hence, we also use a few morphological features derived by the morphological database MorphoLex-EN (Sánchez-Gutiérrez et al., 2018), e.g., number of prefixes, morphemes, and suffixes. We assume the more morphological rich, the more complicated the word.

**Lexicon-based Features.** As, for example, proposed in AbuRa’ed and Saggion (2018), and Wani et al. (2018), we check if the target word is con-

tained in the Oxford 3000 word list (Dictionaries, 2021) with commonly used words. We assume the more common a word is, the simpler it would be.

**Other Features.** Since it is expected that the corpus contains a lot of named entities, such as person names in the bible subcorpus, we check if a target word is a named entity, as also suggested in Gooding and Kochmar (2018).

The last feature is the position of the target word in the sentence. If a target word occurs more than once in a sentence, we consider the word’s last occurrence. In contrast to AbuRa’ed and Saggion (2018), who normalize the word position by the sentence length, we use the absolute word position because we normalize all features afterwards.

## 4.2 Normalization

The hand-crafted features described above all range on different scales, hence, normalizing is required. The normalization is performed as follows: I.) the synsets of the target word are identified, II.) the values of features for every word in the synset are calculated, III.) the values are normalized using min-max normalization. This is being done to compare words that are related to each other, rather than comparing, for instance, frequencies of unrelated words (glee and joyous as opposed to glee and table). In this manner, we are normalizing all the values within a range of 0-1, but by comparing each word with a related word in the synset in which it is present. For words that appear in multiple synsets, we take an average of the normalized values.

As not all features could be normalized relative to their sense (see Table 1), e.g., readability features, we normalized them using scikit-learn’s MinMaxScaler (Pedregosa et al., 2011).

## 4.3 Feature Sets

We create different feature sets considering the normalizing strategies in combination with all character and word embeddings. For the hand-crafted features, we either used the 14 sense relative features, all 34 minmax normalized features, or the 14 sense relative features combined with the missing 20 features minmax normalized (both). All feature sets are listed in Table 2.

## 4.4 Model

RS\_GV’s structure is a more simple version of the structure proposed in De Hertog and Tack (2018), containing linear layers instead of convolutional

layers. Our model is a simple feed-forward neural network with two input layers –one for the hand-crafted features and one for the embedding features–, both followed by a linear hidden layer. Both feature layers are concatenated in another hidden linear layer. It is finally followed by a linear output layer which is activated using the rectified linear unit function (ReLU). We also use stochastic gradient descent (SGD) optimization function. L1Loss as implemented in scikit-learn (Pedregosa et al., 2011) or another mean absolute error loss function seems best for our purpose of predicting continuous labels in a regression task. Following easy stopping, we chose 250 epochs for our model. All hyperparameters with which our model performs best are listed in Appendix A (see Table 5).

RS\_GV can be trained either across all domains at once (*cross-domain*) or on each domain separately (*within-domain*).

## 4.5 Implementation

The system is implemented in Python 3.8 and PyTorch 1.6 (Paszke et al., 2019) using the packages listed in Appendix B (see Table 6). The code of the system is available in our GitHub repository: <https://github.com/gayatrivenugopal/SharedTask-LPC2021>.

## 5 Results

### 5.1 Ablation Tests / Error Analysis

In this section, we report on different approaches made during developing RS\_GV. We compare the results on the trial data using the different feature sets, and a within and a cross domain approach. In the following, we report the average of Pearson correlation on 10 system runs.

#### 5.1.1 Feature Sets

The system’s performance considering all different feature sets is summarized in Table 2.

Embed.	HCF	$r$	SD	Embed.	HCF	$r$	SD
GloVe	sense rel.	0.7654	0.0123	Flair	sense rel.	0.8002	0.0056
GloVe	minmax	0.7721	0.0114	Flair	minmax	0.8007	0.0039
GloVe	both	0.7689	0.0073	<b>Flair</b>	<b>both</b>	<b>0.8027</b>	0.0051
ELMo	sense rel.	0.7648	0.0103	PooledFlair	sense rel.	0.7331	0.0050
ELMo	minmax	0.7667	0.0119	PooledFlair	minmax	0.7685	0.0068
ELMo	both	0.7752	0.0118	PooledFlair	both	0.7537	0.0051
BERT	sense rel.	0.7204	0.0085				
BERT	minmax	0.7260	0.0088				
BERT	both	0.7178	0.0134				

Table 2: Results of all feature sets reporting Pearson correlation  $r$  (average of 10 runs) on the trial data set. The standard deviation is provided in the last column.

**Hand-crafted Feature Sets.** Considering all embedding feature sets (see Table 2), RS\_GV performs often best and with a comparative low standard deviation (see Table 2) with the hand-crafted feature set `both` (e.g.,  $r_{flair}=0.8027, \pm 0.0051$ ) compared to `sense relative` (e.g.,  $r_{flair}=0.8002, \pm 0.0056$ ) and `minmax` (e.g.,  $r_{flair}=0.8007, \pm 0.0039$ ). Hence, in the following, we report the results only on the hand-crafted feature set `both`.

**Embedding Feature Sets.** A comparison between the character embeddings and the word embeddings (see Table 2) shows that `PooledFlair` ( $r=0.7537, \pm 0.0051$ ) could outperform BERT ( $r=0.7178, \pm 0.0134$ ) but `ELMo` could also outperform `PooledFlair` ( $r=0.7752, \pm 0.0118$ ). `Flair` ( $r=0.8027, \pm 0.0051$ ) could outperform all other embedding feature sets.

Surprisingly, RS\_GV with the non-contextualized embeddings feature set (`GloVe`,  $r=0.7689, \pm 0.0073$ ) could outperform all systems with contextualized embeddings except `Flair` ( $r=0.8027, \pm 0.0051$ ) and `ELMo` ( $r=0.7752, \pm 0.0118$ ). It seems that the impact of the contextualization of the embeddings is not as high as expected.

As a compromise of contextualized vs non-contextualized and character vs word embeddings, we use stacked `Flair` embeddings. They combine the forward and backward versions of `Flair` contextualized character embeddings with `GloVe` non-contextualized word embeddings.

### 5.1.2 Cross-domain vs. within-domain

In contrast to the insight of Yimam et al. (2018), RS\_GV performs on average better using the cross-domain approach ( $r=0.8027, \pm 0.0051$ ) than the within-domain approach ( $r=0.7823, \pm 0.0235$ ). The standard deviation of the within-domain approach implies that the model is not as robust as the cross-domain approach. Roughly 3000 instances per domain might be too less to train a robust LCP model with a neural network.

### 5.1.3 Deep Learning vs. Machine Learning

We compare the results of our deep learning approach of RS\_GV with a machine learning regression, i.e., linear regression of scikit-learn. As a result, the neural network and `Flair` ( $r=0.8027, \pm 0.0051$ ) significantly improve LCP compared to the machine learning regression ( $r=0.6945$ ) using only hand-crafted features. Hence, we can confirm the results of the CWI shared task 2018, character embeddings and neural networks do improve LCP.

## 5.2 Submitted Results

Following the previously described ablation tests, we chose to submit the results of the cross-domain approach and the within-domain approach. Both use a deep learning regressor and stacked `Flair` embeddings in combination with the hand-crafted feature set `both`. This section presents the official results of our system RS\_GV on the test set at SemEval 2021 Task 1 sub-task 1 (see also Table 3).

With a Pearson correlation coefficient of  $r=0.7478$  our system with the within-domain outperforms the cross-domain approach on the test data ( $r=0.7316$ ). Officially, RS\_GV ranks on place 34 of 54. The best system proposed by the team *JUST BLUE* achieved  $r=0.7886$ .

Comparing our submitted results with the results on an average of 10 runs (see Table 3), the cross-domain approach can outperform the within-domain approach on the test and trial data.

Overall, both approaches achieve better results than each of the baselines.

Setting or Team	Version	trial $r$	test $r$
within-domain	submission	<b>0.8156</b>	<b>0.7478</b>
cross-domain	submission	0.7978	0.7316
within-domain	average	0.7823	0.7287
cross-domain	average	<b>0.8027</b>	<b>0.7408</b>
Complexity-average	baseline	-	
Length	baseline	0.1589	
Log Frequency	baseline	0.5287	
Log Frequency & Length	baseline	<b>0.5376</b>	
JUST BLUE ()	best team	<b>0.8340</b>	<b>0.7886</b>

Table 3: Results using the trial (3rd) and test dataset (4th column) using Pearson correlation  $r$  for evaluation. The first block contains our submitted and averaged results of 10 runs using `Flair` and `both`. The second block reports the results of the baselines and the third block the results of the best performing system.

## 5.3 Error Analysis

The submitted results reveal that RS\_GV cannot stick with the shared task’s best performing systems. This section presents insights regarding the problems and strengths of RS\_GV on the test data.

**Domain-specific Results.** The subcorpora differ regarding their lexical complexity: The biomed subcorpus has the highest average of lexical complexity in the single word dataset (0.325) and the europarl subset the lowest average (0.286). When we train and predict the lexical complexity per domain,



we can observe the same ranking of the complexity prediction per domain as in [Shardlow et al. \(2020\)](#): The lexical complexity of the europarl domain is most difficult to predict for RS\_GV, whereas the biomedical subcorpus is most easy (see [Table 4](#)).

Domain	Feature set	$r$ (n=10)	SD
all	Flair + both	0.7823	0.0235
bible	Flair + both	0.7177	0.0182
biomed	Flair + both	0.8585	0.0042
europarl	Flair + both	0.7444	0.0089

Table 4: Results of within-domain approach and results per domain using the [trial](#) dataset for evaluation. The Pearson correlation  $r$  is an average of 10 system runs. The standard deviation is provided in the last column.

**High vs. Low Complexity.** It seems that our system can predict complex words better than easy words. However, when splitting the test dataset by complexity value and not by domain, RS\_GV performs poorly on very complex words (complexity value  $> 0.666$ ,  $r=0.0125$ ,  $\pm=0.0542$ ,  $n=12$ ), which might be again due to too less training samples ( $n=105$ ) for the neural network.

Furthermore, the system performs poorly for very easy words (complexity value  $<0.2$ ,  $r=0.0873$ ,  $\pm=0.0272$ ,  $n=188$ ) although roughly 20% of the training samples ( $n=1600$ ) are in this complexity area. We have not found a reason for it yet.

**Homonym-specific Results.** This SemEval task aims at predicting word complexity of tokens in different context including different meanings. Looking more closely on homonyms, on the one hand, different complexity values are assigned to different meanings of a homonym, e.g., vision, but on the other hand, similar complexity values are assigned to a homonym, e.g., resolution. Hence, there is no clear interpretation of how to predict their complexity. This problem is reflected in RS\_GV, our system predicts only slightly different complexity values per homonyms. It seems, that RS\_GV can somehow differentiate the different meaning of the words but overall it differentiate not good enough to perform well in their lexical complexity prediction.

The examples also show the importance of the multi-word LCP task, hence "account" is part of light verb constructions as "to give account" and "to take into account".

**Context-specific Results.** A few samples contain the same token in the (nearly) same sentence, but the complexity values of them are varying (see [Appendix C, Table 9](#)). Removing these 6 out of overall 917 samples of the test data, the system output already improve from 0.7316 to 0.7334. This underlines that LCP is a subjective task and, hence, difficult to predict for machines.

**Linearity.** We tested the data for linearity in order to justify the usage of linear regression. We could not find any linearity between the individual features and the complexity value. The missing linearity might be a reason why RS\_GV could not keep with other systems of the shared task.

## 6 Discussion and Conclusion

We described our model named RS\_GV which was submitted to SemEval Task 2021: Task 1 regarding lexical complexity prediction. We propose a neural network with a combination of hand-crafted and word/character embeddings to approach the task. Our analysis shows that normalization of hand-crafted features using WordNet senses achieves better results than using only a minmax normalization. Furthermore, we figured out that RS\_GV predicts lexical complexity best using a combination of non-contextualized word embeddings and contextualized character embeddings.

In contrast to other shared tasks results, our cross-domain approach achieves better results than the domain-specific approach. A domain-specific approach may need more data to perform reliably.

Furthermore, our neural regressor seems problematic, since it shows some variance in the results on average and the current dataset might be too small for regression with neural networks.

## 7 Future Work

In future works, we plan to improve the character and word embeddings. We could fine-tune the embeddings on our data or use domain-specific pre-trained embeddings, which fits the datasets' domains, e.g., BioFlair ([Sharma and Jr, 2019](#)).

Furthermore, we could calculate more hand-crafted features or edit the current ones. For example, the implementation of sentence readability formulas seems more promising than the misuse of text readability formulas on sentences.

The current neural network contains only a few linear layers, an extension using, e.g., convolutional layers for feature selection seems promising.

## Acknowledgements

We thank the SemEval-2021 Task 1 organizers for preparing and organizing the shared task.

This research is part of the PhD-program "Online Participation", supported by the North Rhine-Westphalian (German) funding scheme "Forschungskolleg".

We gratefully acknowledge NVIDIA Corporation's support with the donation of the Titan Xp GPU used for this research.

## References

- Ahmed AbuRa'ed and Horacio Saggion. 2018. [LaS-TUS/TALN at complex word identification \(CWI\) 2018 shared task](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–165, New Orleans, Louisiana. Association for Computational Linguistics.
- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. [FLAIR: An easy-to-use framework for state-of-the-art NLP](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. [Pooled contextualized embeddings for named entity recognition](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. [Contextual string embeddings for sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Shivam Bansal and Chaitanya Aggarwal. 2014. [Textstat](#). Last updated: 2020-11-20, Last accessed: 2021-02-22.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc."
- Brants, Thorsten and Franz, Alex. 2006. [Web 1t 5-gram version 1](#).
- Meri Coleman and Ta Lin Liao. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60(2):283.
- Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.
- Elnaz Davoodi and Leila Kosseim. 2016. [CLaC at SemEval-2016 task 11: Exploring linguistic and psycho-linguistic features for complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 982–985, San Diego, California. Association for Computational Linguistics.
- Dirk De Hertog and Anaïs Tack. 2018. [Deep learning architecture for complex word identification](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 328–334, New Orleans, Louisiana. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Oxford Learner's Dictionaries. 2021. [Oxford 3000 and 5000](#). Last accessed: 2021-02-22.
- Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.
- Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.
- Sian Gooding and Ekaterina Kochmar. 2019. [Recursive context-aware lexical simplification](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4853–4863, Hong Kong, China. Association for Computational Linguistics.
- Robert Gunning. 1952. *Technique of clear writing*. McGraw-Hill, New York, USA.
- Nathan Hartmann and Leandro Borges dos Santos. 2018. [NILC at CWI 2018: Exploring feature engineering and feature learning](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340, New Orleans, Louisiana. Association for Computational Linguistics.
- Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

- Colby Horn, Cathryn Manduca, and David Kauchak. 2014. [Learning a lexical simplifier using Wikipedia](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, Maryland. Association for Computational Linguistics.
- David Kauchak. 2013. [Improving text simplification language modeling using unsimplified text data](#). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria. Association for Computational Linguistics.
- J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.
- Gustavo Paetzold and Lucia Specia. 2016. [SemEval-2016 task 11: Complex Word Identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [PyTorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A python natural language processing toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.
- Claudia H Sánchez-Gutiérrez, Hugo Mailhot, S Hélène Deacon, and Maximiliano A Wilson. 2018. [MorphoLex: A derivational morphological database for 70,000 english words](#). *Behavior research methods*, 50(4):1568–1580.
- Toby Segaran and Jeff Hammerbacher. 2009. *Beautiful data: The stories behind elegant data solutions*, 1 edition. O’Reilly Media, Inc.
- RJ Senter and Edgar A Smith. 1967. Automated readability index. Technical report, Cincinnati University, Cincinnati, USA.
- Matthew Shardlow. 2013. [The CW corpus: A new resource for evaluating the identification of complex words](#). In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 69–77, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow. 2014. [Out in the open: Finding and categorising errors in the lexical simplification pipeline](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 1583–1590, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex: A new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021a. [SemEval-2021 Task 1: Lexical Complexity Prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. [Predicting lexical complexity in english texts](#). *arXiv preprint arXiv:2102.08773*.
- Shreyas Sharma and Ron Daniel Jr. 2019. [BioFLAIR: Pretrained pooled contextualized embeddings for biomedical sequence labeling tasks](#). *arXiv preprint arXiv:1908.05760*.
- Serge Sharoff. 2006. [Open-source corpora: Using the net to fish for linguistic data](#). *International Journal of Corpus Linguistics*, 11(4):435–462.

Rachael Tatman. 2017. [English word frequency: 1/3 million most frequent english words on the web](#). Last updated: 2018-09-28; Last accessed: 2021-02-22.

Nikhil Wani, Sandeep Mathias, Jayashree Aanand Gajjam, and Pushpak Bhattacharyya. 2018. [The whole is greater than the sum of its parts: Towards the effectiveness of voting ensemble classifiers for complex word identification](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 200–205, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Wilson. 1988. Mrc psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior research methods, instruments, & computers*, 20(1):6–10.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. [CWIG3G2 - complex word identification task across three text genres and two user groups](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.

## A Hyperparameter

Hyperparameter	Final Value	Fine-tuning Values
classifier	Neural Lin. Regression	sklearn LinearSVR, sklearn LinearRegression, Neural Lin. Regression
learning rate	0.075	0.01, 0.05, 0.075, 0.1, 0.2
epochs	250	100, 250, 500
# input layer	2	1, 2
# hidden layer	3	1, 2, 3
hidden size (HCF)	128	128, 256, 512
hidden size (EM)	256	256, 512, 1024
hidden size (concat)	128	128, 256, 512
criterion	L1Loss	L1Loss, MSELoss, SmoothL1Loss
optimizer	SGD	SGD, ADAM
dropout	-	0.1, 0.05, 0.01

Table 5: Hyperparameter during fine tuning and the final chosen hyperparameter of the proposed system.

## B Resources

### B.1 Python Packages

Package	Usage	Package	Usage
pandas	Data Import	torch	Regression
xldr	Data Import	scikit-learn	Regression
spacy	Preprocessing	interpret	Regression
stanza	Preprocessing	numpy	Ablation Study & Error Analysis
nltk	WordNet Feature	seaborn	Data Visualization
syllables	Syllable Feature	yellowbrick	Data Visualization
textstat	Readability Feature	visdom	Data Visualization
Flair	Embedding Feature		
torch	Model		
scikit-learn	Evaluation		

Table 6: Python packages used for the implementation of the proposed system.

## B.2 Language Resources

name	usage	type	access	reference
CompLex	data	corpus	<a href="https://github.com/MMU-TDMLab/CompLex">https://github.com/MMU-TDMLab/CompLex</a>	Shardlow et al. (2020, 2021b)
Textstat	readability features	python package	<a href="https://pypi.org/project/textstat/">https://pypi.org/project/textstat/</a>	Bansal and Aggarwal (2014)
internet-en-forms	frequency feature	word list	<a href="http://corpus.leeds.ac.uk/list.html#frqc">http://corpus.leeds.ac.uk/list.html#frqc</a>	Sharoff (2006)
GoogleWeb1T, unigram_freq.csv	frequency feature	word list	<a href="https://www.kaggle.com/rtatman/english-word-frequency">https://www.kaggle.com/rtatman/english-word-frequency</a>	Tatman (2017)
GoogleWeb1T, count_1w.txt	frequency feature	word list	<a href="https://norvig.com/ngrams/">https://norvig.com/ngrams/</a>	Segaran and Hammerbacher (2009)
NLTK	lexical feature	NLP library	<a href="https://www.nltk.org/">https://www.nltk.org/</a>	Bird et al. (2009)
spaCy	lexical feature	NLP library	<a href="https://github.com/explosion/spaCy">https://github.com/explosion/spaCy</a>	Honnibal and Montani (2017)
stanza	lexical feature	NLP library	<a href="https://stanfordnlp.github.io/stanza/">https://stanfordnlp.github.io/stanza/</a>	Qi et al. (2020)
syllapy	lexical feature	python package	<a href="https://github.com/mholtzscher/syllapy">https://github.com/mholtzscher/syllapy</a>	
syllable	lexical feature	python package	<a href="https://github.com/prosegrinder/python-syllables">https://github.com/prosegrinder/python-syllables</a>	
WordNet	WordNet feature	database	<a href="https://www.nltk.org/api/nltk.corpus.reader.html?highlight=wordnet#module-nltk.corpus.reader.wordnet">https://www.nltk.org/api/nltk.corpus.reader.html?highlight=wordnet#module-nltk.corpus.reader.wordnet</a>	Fellbaum (1998)
Oxford 3000	lexicon feature	word list	<a href="https://github.com/gokhanyavas/Oxford-3000-Word-List/blob/master/Oxford%203000%20Word%20List%20No%20Spaces.txt">https://github.com/gokhanyavas/Oxford-3000-Word-List/blob/master/Oxford%203000%20Word%20List%20No%20Spaces.txt</a>	Dictionaries (2021)
MorphoLex-EN	morphological feature	database	<a href="https://github.com/hugomailhot/MorphoLex-en">https://github.com/hugomailhot/MorphoLex-en</a>	Sánchez-Gutiérrez et al. (2018)
MRC Psycholing. Database	psycholinguistic feature	database	<a href="https://github.com/samzhang111/mrc-psycholinguistics/raw/master/mrc2.dct">https://github.com/samzhang111/mrc-psycholinguistics/raw/master/mrc2.dct</a>	Wilson (1988)
FLAIR	embedding feature	NLP framework	<a href="https://github.com/flairNLP/flair">https://github.com/flairNLP/flair</a>	Akbik et al. (2019a)
GloVe	word embedding feature	pretrained embeddings	<a href="http://nlp.stanford.edu/data/glove.6B.zip">http://nlp.stanford.edu/data/glove.6B.zip</a>	Pennington et al. (2014)
FLAIR	character embedding feature	pretrained embeddings	<a href="https://github.com/flairNLP/flair">https://github.com/flairNLP/flair</a>	Akbik et al. (2018, 2019a)
PooledFlair	character embedding feature	pretrained embeddings	<a href="https://github.com/flairNLP/flair">https://github.com/flairNLP/flair</a>	Akbik et al. (2019b)
BERT	word embedding feature	pretrained embeddings	<a href="https://github.com/google-research/bert">https://github.com/google-research/bert</a>	Devlin et al. (2019)
ELMo	word embedding feature	pretrained embeddings	<a href="https://allennlp.org/elmo">https://allennlp.org/elmo</a>	Peters et al. (2018)

Table 7: All used language resources listed with usage, access and reference.

## B.3 Word and Character Embeddings

embedding name	type	context	specification	domain	corpora	dimensions
Flair	character	x	Mix-forward, mix-backward, glove	web, wikipedia, subtitles	1 Billion Word Benchmark	4196
PooledFlair	character	x	Mix-forward	web, wikipedia, subtitles	-	4096
BERT	word	x	bert-base-uncased	Fiction, news, wikipedia	BooksCorpus, Wikipedia, 1 Billion Word Benchmark	3072
ELMO	word	x	original	news	1 Billion Word Benchmark	3072
GloVe	word		glove.6B.300d	wikipedia, news	Wikipedia 2014, Gigaword 5	300

Table 8: Settings of the word and character embeddings.

## C Detailed Results

### C.1 Context-specific Results.

ID	Sentence	Token	Complexity	Predicted
39HYCOOPKOL434K1UCPA8CBZRO4DMM	Arguably, since the body pools and plasma sitosterol levels in the knockout mice are so considerably elevated, perhaps the biliary sitosterol levels could be considered to be inappropriately low.	body	0.0499	0.2049
3ZZAYRN1I6RZKW1ATI425KIQA7TO0	Arguably, since the body pools and plasma sitosterol levels in the knockout mice are so considerably elevated, perhaps the biliary sitosterol levels could be considered to be inappropriately low.	body	0.3173	0.2049
3KWGG5KP6J2UYCENUGUZO6TH6QDCMA	Fishing opportunities and financial contribution provided for by the EU-Seychelles Fisheries Partnership Agreement (	Fisheries	0.3088	0.3387
341H3G5YF0EA3RIQXPR917NPC4EZ0Z	Fishing opportunities and financial contribution provided for in the EU-São Tomé and Príncipe Fisheries Partnership Agreement (	Fisheries	0.1875	0.3405
3LCXHSGDLT6CT5B6A4WGQ3SQJNDSSES	Therefore thus says Yahweh of Armies concerning the prophets: Behold, I will feed them with wormwood, and make them drink the water of gall; for from the prophets of Jerusalem is ungodliness gone forth into all the land.	wormwood	0.7321	0.4117
3DWNFENNE3V120VNY4BPPGPAHX4JD	therefore thus says Yahweh of Armies, the God of Israel, Behold, I will feed them, even this people, with wormwood, and give them water of gall to drink.	wormwood	0.4843	0.4170

Table 9: Samples of the test set with the same token in the same sentence but different complexity values. The last column contains the predicted values of RS\_GV .

# UNBNLP at SemEval-2021 Task 1: Predicting lexical complexity with masked language models and character-level encoders

Milton King and Ali Hakimi Parizi and Samin Fakharian and Paul Cook

Faculty of Computer Science, University of New Brunswick

Fredericton, NB E3B 5A3, Canada

{milton.king, ahakimi, samin.fakharian, paul.cook}@unb.ca

## Abstract

In this paper, we present three supervised systems for English lexical complexity prediction of single and multiword expressions for SemEval-2021 Task 1. We explore the use of statistical baseline features, masked language models, and character-level encoders to predict the complexity of a target token in context. Our best system combines information from these three sources. The results indicate that information from masked language models and character-level encoders can be combined to improve lexical complexity prediction.

## 1 Introduction

SemEval 2021 Task 1 (Shardlow et al., 2021) focuses on predicting the complexity of a target token in an English sentence on a scale from 0 (not very complex) to 1 (very complex). For example, the token *land* in the example below is judged to have a low complexity of 0.19.

1. Our land will yield its increase.

On the other hand, the token *doxycycline* in the following example is judged to have a relatively higher complexity of 0.75.<sup>1</sup>

2. The reason these two lines were unresponsive to doxycycline is unknown.

The dataset for this task was originally proposed in Shardlow et al. (2020), and includes sentences from three sources: a translated bible, European Parliament proceedings, and a biomedical corpus. This shared task contains two sub-tasks with the first focusing on lexical complexity for single words,

<sup>1</sup>These example sentences and complexity scores are taken from the dataset provided for the shared task.

which will be referred to as *SINGLE*, and the second focusing on complexity of multiword expressions, which will be referred to as *MULTI*.

In this paper, we explore the use of statistical baseline features, masked language models, and character-level encoders to predict the complexity of a target token in context. We first consider these approaches individually and then consider supervised methods for combining them. We evaluate our models with Pearson correlation (R), Spearman correlation (Rho), mean absolute error (MAE), mean square error (MSE), and R-squared (R2). We apply all our models to both sub-tasks and find that we achieve our best results with a model that combines all three sources of information — baseline features, masked language modeling, and character-level encoding. Specifically, we achieve our best results with respect to Pearson correlation — the evaluation measure used for the official shared task system ranking — using a model that combines complexity predictions from two approaches, one based on a character-level encoder, and the other based on a masked language model, which further incorporates the baseline features, using support vector regression. Although we achieve our best results using this approach on both *SINGLE* and *MULTI* with respect to Pearson correlation, we note substantial variation in performance with respect to the other evaluation metrics on *SINGLE*. This suggests that future work could further explore the variation in performance of this model on single and multiword expressions

## 2 Model components

In this section, we describe three approaches to lexical complexity prediction. In Section 3 we then describe how these approaches are combined into systems that we submitted as official runs to the shared task.

## 2.1 Baseline features

We extract a set of seven statistical baseline features. These features include the length of the target token, the frequency of the token in SubtlexUS (Brysbaert and New, 2009), the frequency of the token in SubtlexUK (van Heuven et al., 2014), a binary feature indicating whether the token is an MWE, and a binary feature for each text type the instances were taken from (i.e., biomedical text, European Parliament proceedings, and the bible) indicating whether the instance is from that corpus. For MWEs, the frequency features are calculated as the average of the frequencies of the component words in the MWE. In the case that frequency information is not available for a word in SubtlexUS or SubtlexUK, the frequency of that word is set to the average frequency for words in the dataset for which frequency information is available. These features are combined with the approach described in Section 2.3, and the systems presented in Section 3.

## 2.2 Character-level encoder

This approach is based on a character-level encoder, which has three parts. The first part is a pre-trained character-level language model which gets a sentence containing a target expression as its input and its last hidden state is used as an embedding representing the input sentence. This language model uses a bi-directional GRU with a hidden layer size of 256. It is trained on the sentences in the trial and training data provided for the shared task.

The second part is a similar pre-trained GRU bi-directional character-level language model, which receives the target word as its input and the first hidden state is initialized with the embedding of the input sentence. The last hidden state is then passed to the systems described in Section 3 for complexity prediction.

In this approach, our hypothesis is that the hidden state of the first language model provides a representation for the input sentence, and the language model can encode the complexity of the target word by having access to a representation of the sentence in which the target word appears. Figure 1 shows a diagram of this model.

## 2.3 Masked language model

In this approach, we recruit BERT (Devlin et al., 2019) as a masked language model to estimate the probability for the target token in context. Collins-

Thompson and Callan (2004) show that language modeling can be used to predict reading difficulty, and therefore we hypothesize that language modeling can also be useful for predicting lexical complexity.

We use the large uncased pretrained BERT model, which consists of 16 heads and 24 layers of 1024 hidden units each. Given a sentence, we replace the target token with the special *[MASK]* token and use the modified sentence as input to BERT to obtain the probability of the target token. BERT’s tokenizer can split tokens into multiple pieces. The probability of a target token (single word or multiword expression) is therefore calculated by averaging the probabilities of its parts.

The probability of the target is then used as a feature, alongside the baseline features from Section 2.1, in a support vector regressor (SVR) to predict complexity. The SVR uses an rbf kernel, with a kernel coefficient of 0.1, epsilon of 0.1, and a regularization of 1/100. The output of the SVR is used as a feature in the systems described in Section 3. Figure 2 shows a diagram of this model.

## 3 Submitted Systems

In this section, we describe how we combine the approaches discussed in Section 2 to form systems that were submitted as official runs to the shared task. We selected these systems because they achieved the best performance in a ten-fold cross-validation experiment on the combined trial and training data, in terms of Pearson correlation (R), the evaluation metric used to rank systems in the shared task. The performance of each submitted system, a baseline in which we train logistic regression on the baseline features from Section 2.1, and the masked language model approach described in Section 2.3 (our best-performing model that was not submitted to the shared task), are shown in Tables 1 and 2 for *SINGLE* and *MULTI*, respectively.

### 3.1 System 1

In this system, we concatenate the output of the character-level encoder approach discussed in Section 2.2 with the baseline features from Section 2.1. This representation is then used as input to a feed-forward network to predict the complexity. The feed-forward network has two fully connected hidden layers with sizes 128 and 64, and ReLU activation functions. We train this network using Adam optimizer (Kingma and Ba, 2015) with a

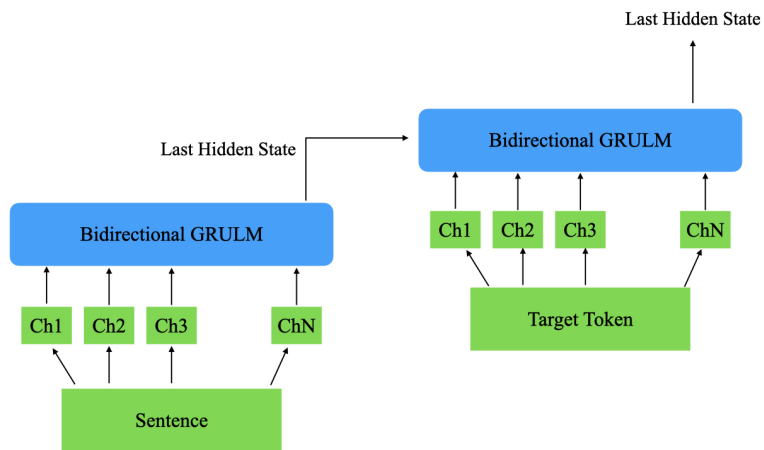


Figure 1: The character-level encoder model.

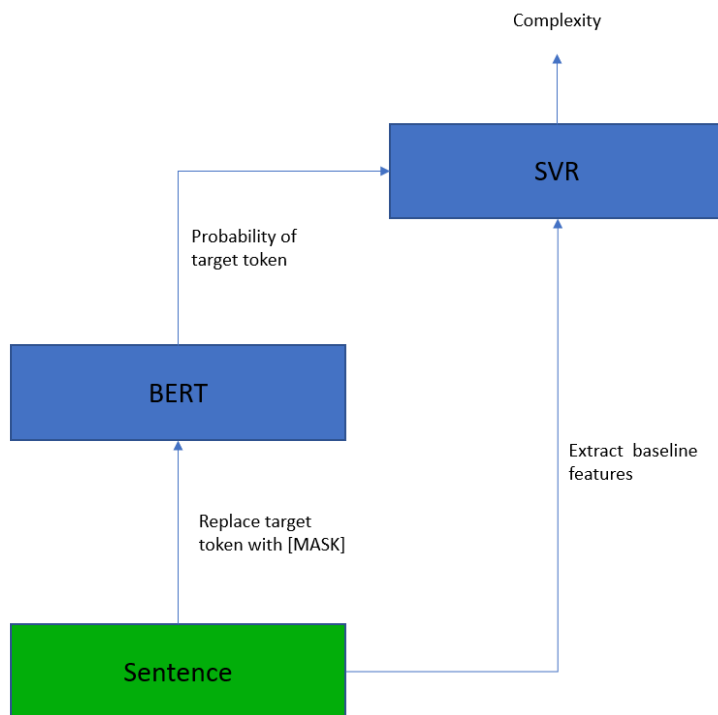


Figure 2: A diagram of the masked language model.



Model	R	Rho	MAE	MSE	R2
<i>Baseline</i>	0.618	0.633	0.080	0.011	0.378
<i>MLM</i>	0.622	0.635	0.080	0.011	0.383
<i>System<sub>1</sub></i>	0.698	0.673	0.074	0.009	0.476
<i>System<sub>2</sub></i>	0.712	0.680	0.072	0.009	0.499
<i>System<sub>3</sub></i>	0.705	0.678	0.073	0.009	0.482

Table 1: Results for each model for each evaluation metric on single word expressions in the validation set. *MLM* is the masked language model described in Section 2.3. (For R, Rho, and R2 bigger numbers indicate better performance, whereas for MSE and MAE, smaller numbers are better.)

Model	R	Rho	MAE	MSE	R2
<i>Baseline</i>	0.665	0.683	0.094	0.015	0.412
<i>MLM</i>	0.665	0.684	0.094	0.015	0.411
<i>System<sub>1</sub></i>	0.675	0.686	0.092	0.014	0.440
<i>System<sub>2</sub></i>	0.684	0.693	0.091	0.014	0.442
<i>System<sub>3</sub></i>	0.673	0.684	0.091	0.014	0.432

Table 2: Results for each model for each evaluation metric on multiword expressions in the validation set.

learning rate of 0.001 for 100 epochs. This system is referred to as *System<sub>1</sub>* from heron.

### 3.2 System 2

In this system, we concatenate the feature from the masked language model approach described in Section 2.3 with the output from the system described in Section 3.1 to create a 2-dimensional vector. This vector is then used as input to an SVR to predict complexity.

We perform a grid search to tune the hyperparameters of the SVR via evaluating its performance on the validation set. We achieve our best results with an SVR using a polynomial kernel with a degree of 3, a kernel coefficient of 1, a regularization parameter of 100, a stopping tolerance of 1, and a gamma value of  $1/\text{number of training instances}$ . We use these parameters for all further experiments with this system, which we refer to *System<sub>2</sub>*.

### 3.3 System 3

In this system, similar to *System<sub>1</sub>*, we concatenate the output of several approaches from Section 2, and then pass this representation to a fully connected network to predict the complexity. Here we concatenate the baseline features with the output of both the character-level encoder approach (Section 2.2) and the masked language model approach (Section 2.3). We use the same fully-connected net-

Model	R	Rho	MAE	MSE	R2
<i>Baseline</i>	0.584	0.597	0.080	0.108	0.334
<i>System<sub>1</sub></i>	0.691	<b>0.656</b>	0.073	0.0094	0.418
<i>System<sub>2</sub></i>	<b>0.695</b>	0.654	0.072	0.0089	0.450
<i>System<sub>3</sub></i>	0.689	0.653	<b>0.069</b>	<b>0.0086</b>	<b>0.471</b>

Table 3: Results on *SINGLE* for each system and each evaluation metric. The best result for each evaluation metric is shown in boldface.

Model	R	Rho	MAE	MSE	R2
<i>Baseline</i>	0.731	0.704	0.092	0.013	0.470
<i>System<sub>1</sub></i>	0.741	0.735	0.0843	0.0116	0.519
<i>System<sub>2</sub></i>	<b>0.752</b>	<b>0.742</b>	<b>0.0802</b>	<b>0.0106</b>	<b>0.562</b>
<i>System<sub>3</sub></i>	0.736	0.730	0.0851	0.0116	0.521

Table 4: Results on *MULTI* for each system and each evaluation metric. The best result for each evaluation metric is shown in boldface.

work structure, and training settings, as for *System<sub>1</sub>*. We refer to this approach as *System<sub>3</sub>*.

## 4 Results

In this section we present our results with respect to the five evaluation metrics. We evaluate our models on the single word sub-task (*SINGLE*) first and then on the multiword expressions sub-task (*MULTI*). Each system is trained on all training instances from both *SINGLE* and *MULTI*.<sup>2</sup> We include the performance of our baseline to show that all submitted systems continue to outperform this baseline model on the test data.

In Table 3, we show the performance of our systems on *SINGLE*. *System<sub>2</sub>* achieves the best results with respect to *R*, which is the metric used to rank submissions in the shared task. Interestingly, however, this is the only metric for which *System<sub>2</sub>* outperforms the other systems. *System<sub>3</sub>* — which like *System<sub>2</sub>* incorporates information from both the character-level encoder and masked language model approaches — performs worst of these three systems with respect to *R*, but achieves the best performance amongst these systems for MSE, MAE, and R2.

In Table 4, we show the performance of our models on *MULTI*. In contrast to the results on *SINGLE*, these results show *System<sub>2</sub>* consistently performs

<sup>2</sup>The training data for the approach described in Section 2.3 also includes instances from the provided trial data from both *SINGLE* and *MULTI*.

best for all evaluation metrics. The improvement of *System*<sub>2</sub> — which uses information from both the character-level encoder and masked language model approaches — over *System*<sub>1</sub> in particular — which does not incorporate information from the masked language model — suggests that these two sources of information can be combined to improve lexical complexity prediction.

## 5 Conclusions

We evaluated three systems for lexical complexity prediction of single and multiword expressions for SemEval 2021 Task 1. These systems incorporated information from statistical baseline features, a character-level encoder approach, and a masked language model approach. We found that a system that combined the complexity predictions of the character-level encoder approach and the masked language model approach, which further incorporates the statistical baseline features, using support vector regression performed best amongst our submitted systems with respect to Pearson correlation on both the single word and multiword expressions sub-tasks. This approach further performed best of our submitted systems with respect to all evaluation metrics on the multiword expression sub-task, although this was not the case for the single word sub-task.

In future work, the relationship between the sub-tasks, models, and evaluation metrics warrants further exploration, including studying the effect that the type of the target expression, i.e., single word or multiword expression — has on the performance of the models with respect to the various evaluation metrics.

## Acknowledgments

This work is financially supported by the Natural Sciences and Engineering Research Council of Canada and the University of New Brunswick.

## References

- M. Brysbaert and B. New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior Research Methods*, 41:977–990.
- Kevyn Collins-Thompson and James P. Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the Human Lan-*

*guage Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 193–200, Boston, Massachusetts, USA. Association for Computational Linguistics.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

- Walter van Heuven, Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. Subtlex-uk: a new and improved word frequency database for british english. *Quarterly journal of experimental psychology (2006)*, 67.

- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.

- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

# ANDI at SemEval-2021 Task 1: Predicting complexity in context using distributional models, behavioural norms, and lexical resources

Armand Rotaru  
Independent researcher  
armand.rotaru@gmail.com

## Abstract

In this paper we describe our participation in the Lexical Complexity Prediction (LCP) shared task of SemEval 2021, which involved predicting subjective ratings of complexity for English single words and multi-word expressions, presented in context. Our approach relies on a combination of distributional models, both context-dependent and context-independent, together with behavioural norms and lexical resources.

## 1 Introduction

In our day-to-day life, outside the laboratory, we almost never come across single words or pairs of words, in isolation. Instead, such verbal stimuli are typically embedded within sentences or phrases, and our understanding of individual words and word pairs is influenced by their linguistic contexts (e.g., by disambiguating their intended meaning). However, almost all behavioural norms collected so far focus only on single words or word pairs (Johns, Jamieson, & Jones, 2020).

Therefore, the Lexical Complexity Prediction (LCP) shared task (Shardlow, Evans, Paetzold, & Zampieri, 2021), hosted at SemEval 2021, constitutes a timely and valuable contribution to the study of context-dependent semantics. The task requires competitors to predict subjective ratings of complexity for words or pairs of words, presented within sentences. As mentioned by the organisers, being able to automatically estimate contextualised complexity ratings would have several practical applications, such as detecting and simplifying portions of text that might be particularly difficult

to understand for second language learners, and people with low literacy levels (e.g., as a result of suffering from a reading impairment).

The dataset for the competition is CompLex 2.0 (Shardlow, Cooper, & Zampieri, 2020; Shardlow, Evans, & Zampieri, 2021), consisting of passages from the Bible, the proceedings of the European Parliament, and biomedical journal articles. The training data covers 7,662 single words (2,574 bible, 2,512 europarl, and 2,576 biomed), and 1,517 multi-word expressions (505 bible, 498 europarl, and 514 biomed). The test data covers 917 single words (283 bible, 345 europarl, and 289 biomed), and 184 multi-word expressions (66 bible, 65 europarl, and 53 biomed).

In this paper we describe our submission to the competition, based on distributional models, both context-dependent and context-independent, as well as behavioural norms/lexical resources<sup>1</sup>. The best results are obtained by combining the three classes of predictors. However, the improvement in performance over using just context-independent models is small, and, in practice, might be compensated by their impressive vocabulary size and ease of use.

## 2 General Description

In order to predict word complexity in context, we combined information from three type of sources, namely behavioural norms/lexical resources, and distributional models. With respect to the latter, we included two distinct classes of models:

- context-independent models, which output the same vector representation for a given word, regardless of the context in which the word is encountered;

---

<sup>1</sup> <https://github.com/armandrotaru/TeamAndi-LCP>

- context-dependent models, which output a potentially different representations for a given word, as a function of the context in which the word is presented.

Our approach was very similar to that employed in (Rotaru, 2020), for predicting ratings of concreteness in context.

Firstly, we used behavioural norms collected for a wide variety of psycholinguistic factors, as well as lexical resources. More specifically, we focused on norms for concreteness (Brysbaert, Warriner, & Kuperman, 2014; Paetzold & Specia, 2016), imageability (Paetzold & Specia, 2016), semantic diversity (Hoffman, Lambon Ralph, & Rogers, 2013), age of acquisition (Kuperman, Stadthagen-Gonzalez, & Brysbaert, 2012; Paetzold & Specia, 2016), familiarity (Paetzold & Specia, 2016), emotional dimensions (i.e., valence, arousal, and dominance; Mohammad, 2018), and sensorimotor dimensions (i.e., modality strengths for the tactile, auditory, olfactory, gustatory, visual, and interoceptive modalities; interaction strengths for the mouth/throat, hand/arm, foot/leg, head excluding mouth/throat, and torso effectors; Lynott, Connell, Brysbaert, Brand, & Carney, 2019). We also included complexity ratings (Maddela & Xu, 2018), lexical decision response times and accuracies (Keuleers, Lacey, & Brysbaert, 2012), contextual diversity counts (Van Heuven, Mandera, Keuleers, & Brysbaert, 2014), frequency counts (Van Heuven et al., 2014; Lin et al. 2012), prevalence counts (Brysbaert, Mandera, McCormick, & Keuleers, 2019), and CEFR word lists (Council of Europe, 2001). Nearly all these measures are correlated with word complexity.

Secondly, we employed context-independent distributional models, namely Skip-gram (Mikolov, Chen, Corrado, & Dean, 2013), GloVe (Pennington, Socher, & Manning, 2014), and ConceptNet NumberBatch (Speer, Chin, & Havasi, 2017). Such models have been used in order to accurately predict a range of psycholinguistic variables (e.g., Hollis, Westbury, & Lefsrud, 2017; Utsumi, 2020), which suggests that they could be useful in accounting for complexity ratings.

Thirdly, we employ context-dependent distributional models, namely BERT (Devlin, Chang, Lee, & Toutanova, 2019), RoBERTa (Liu et al., 2019), ELECTRA (Clark, Luong, Le, & Manning, 2020), ALBERT (Lan et al., 2020), and DeBERTa (He, Liu, Gao, & Chen, 2020). Given that such models achieve human-level

performance in various linguistic tasks (e.g., those in the GLUE benchmark; Wang et al., 2018), and that they were designed to process rich contextual information, they could be a valuable tool for predicting ratings of complexity in context.

### 3 System Description

We tested three groups of predictors, both in isolation and combined. The first group was obtained from comprehensive datasets of subjective ratings (concreteness, age of acquisition, etc.), task performance measures (i.e., response times and accuracies in the lexical decision tasks), as well as frequency, contextual diversity, and prevalence counts, plus CEFR word lists (see the references from the beginning of the previous section). In order to extend the coverage of the subjective ratings, we did not use the original data, but instead relied on extrapolated ratings for more than 70,000 words. The extrapolation was based on the Skip-gram, GloVe, and ConceptNet NumberBatch models, using linear regression over the concatenated vector dimensions. For the (already extrapolated) ratings from (Paetzold & Specia, 2016), as well as for the frequency, contextual diversity, and prevalence counts, we employed only the normed values, as they already have very good coverage. We also used only the original lexical decision data, given that response times and accuracies are difficult to extrapolate, and did not try to extend the CEFR word lists, due to methodological difficulties. For the single word datasets, we employed all the previously mentioned factors, whereas for the multi-word expression datasets, we only employed our own extrapolated factors.

The second group was generated from Skip-gram, GloVe, and ConceptNet NumberBatch embeddings. The vocabulary of the models was that described in the discussion above.

For the first two sources of information, and for each selected variable  $V$  (e.g., semantic diversity), we generated either four predictors, in the case of the single word datasets, or nine predictors, in the case of the multi-word expression datasets. The single word predictors consisted of  $V(w)$ ,  $V(c)$ ,  $V(w) * V(c)$ , and  $\text{abs}(V(w) - V(c))$ , while the multi-word expression predictors consisted of  $V(w_1)$ ,  $V(w_2)$ ,  $V(c)$ ,  $\text{abs}(V(w_1) - V(c))$ ,  $\text{abs}(V(w_2) - V(c))$ ,  $\text{abs}(V(w_1) - V(w_2))$ ,  $V(w_1) * V(c)$ ,  $V(w_2) * V(c)$ ,  $V(w_1) * V(w_2)$ , where:

- $V(w)$  denotes the value of  $V$  corresponding to the single word  $w$  (e.g.,  $w = \text{“sons”}$ ). If  $w$  is not present in our norms/models, we set  $V(w)$  to the average value of  $V$ , computed over the entire vocabulary;
- $V(w_1)$  and  $V(w_2)$  denote the values of  $V$  corresponding to the words  $w_1$  and  $w_2$  (e.g.,  $w_1 = \text{“skillful”}$ ,  $w_2 = \text{“workman”}$ ), that make up the multi-word expression  $w_1 w_2$  (i.e.,  $w_1 w_2 = \text{“skillful workman”}$ ). As before, if  $w_1$  and/or  $w_2$  are not present in our norms/models, we set  $V(w_1)$  and/or  $V(w_2)$  to the average value of  $V$ , computed over the entire vocabulary;
- $V(c)$  denotes the value of  $V$  corresponding to the context  $c$  in which the single word  $w$ , or multi-word expression  $w_1 w_2$ , are encountered (e.g.,  $w = \text{“sons”}$ ,  $c = \text{“The ____ of Perez: Hezron, and Hamul.”}$ ; or  $w_1 w_2 = \text{“skillful workman”}$ ,  $c = \text{“He made it the work of a ____.”}$ ). Computing this value involves calculating the average  $V(c) = \frac{\sum_{i=1}^N V(c_i)}{N}$ , where  $V(c_i)$  is the value of  $V$  corresponding to the  $i$ -th context word, calculated as described previously, and  $N$  is the number of context words.

These predictors allowed us to include both the individual contributions of the single word  $w$ , or the multi-word expression  $w_1 w_2$ , and the context  $c$ , as well as certain interactions between the former and the latter.

The third group was derived from the BERT, RoBERTa, ELECTRA, ALBERT, and DeBERTa models. We used the standard (base) versions of each model (i.e., without task-specific fine-tuning), as described in the original papers, with the exception of ELECTRA, where we employed the small, base, and large versions of the model. The implementations of the models were all obtained from the Hugging Face repository (Wolf et al., 2020). The predictors consisted only of the activations for the single word  $w$ , or the multi-word expression  $w_1 w_2$ , averaged over the last four hidden layers.

To predict ratings of complexity in context, we employed ridge regression ( $\lambda = 3000$ ), for the single word dataset, and a combination of ridge regression ( $\lambda = 1250$ ) and gradient-boosted decision trees, for the multi-word expression dataset, after zero centering all the aforementioned predictors.

## 4 Results and Discussion

The results for English are shown Figure 1, for various sets of predictors and regularization strengths. For reasons of space, we only present the results for ridge regression, but note that similar patterns of performance are obtained for gradient-boosted decision trees and other types of models, such as shallow neural networks. Results are averaged over 10 rounds of 10-fold cross-validation, using only the training dataset.

The results indicate that context-independent models (Fig. 1b) outperform behavioural norms (Fig. 1a), and context-dependent models (Fig. 1c-f). A likely reason for the superiority of context-independent models over context-dependent models is the fact that the former were trained on huge corpora (i.e., 100-840 billion tokens), while the latter were trained on considerably smaller corpora (i.e., 3-33 billion tokens). However, in spite of this significant training disadvantage, context-dependent models produce competitive levels of performance, a finding which can likely be attributed to several factors, such as the highly non-linear integration of contextual information, the use of self-attention mechanisms, and that of more sophisticated learning objectives.

Combining the three classes of predictors produces a relatively small improvement in predictive performance, as compared to relying on any single class. This reflects a very high degree of redundancy between the complexity-related information present in the three types of predictors.

Interestingly, even for the largest set of predictors, consisting of 13,400 variables per 1,517 data points, the degree of regularization does not appear to matter much, indicating little overfitting.

Finally, there is a small, but systematic difference in performance between single words and multi-word expressions, in favour of the latter, even though the training set for single word stimuli is roughly five times larger than that for multi-word stimuli. A potential explanation for this finding might be that the individual variability in meaning for multi-word expressions is smaller than that for single words, given that expressions should be more informative than single words, in virtue of their length (i.e., two words vs one word).

Within the competition, our models ranked 4<sup>th</sup> ( $r = .78$ ,  $\rho = .73$ ,  $MAE = .064$ ), in the single word sub-task, and 6<sup>th</sup> ( $r = .85$ ,  $\rho = .84$ ,  $MAE = .067$ ), in the multi-word expression sub-task.

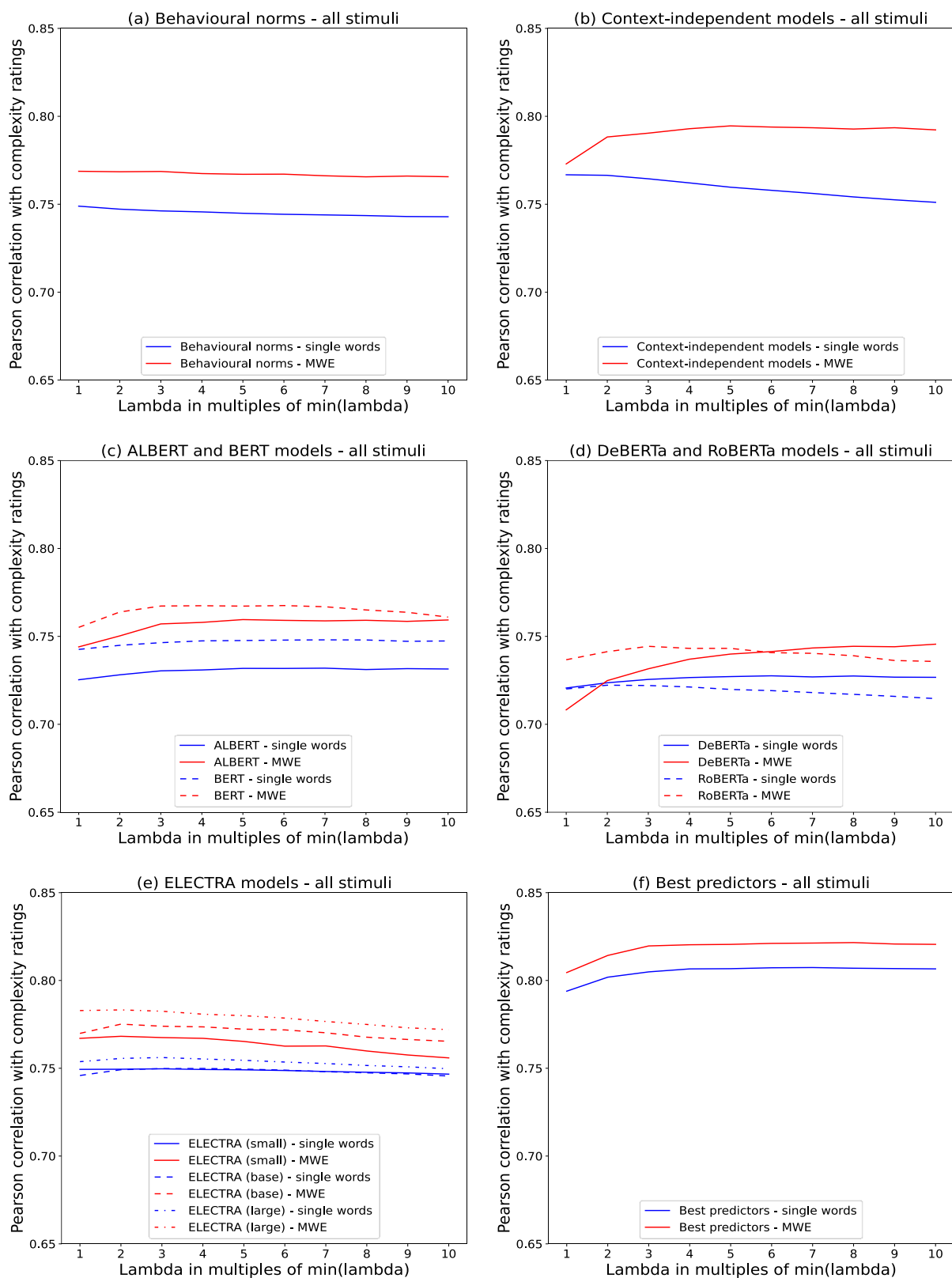


Figure 1. Pearson correlations between predicted and actual complexity ratings, for various groups of predictors and regularization strengths (i.e., values of  $\lambda$ ). For single words, in subfigures (a)-(e),  $\min(\lambda) = 100$ , while in subfigure (f),  $\min(\lambda) = 400$ . For multi-word expressions, in subfigures (a)-(e),  $\min(\lambda) = 50$ , while in subfigure (f),  $\min(\lambda) = 200$ .

## 5 Conclusions

Our results suggest that several approaches can be quite successfully employed in order to predict ratings of complexity in context, for both single words and multi-word expressions. In terms of performance, the best predictors are those derived from context-independent models (e.g., Skip-gram), but relatively good results can be obtained also by using context-dependent models (e.g., BERT) and behavioural norms (e.g., subjective ratings of familiarity). Moreover, given that their vocabulary covers a remarkable number of words (i.e., more than 500 thousand, for each of the Skip-gram, GloVe, and ConceptNet NumberBatch models), and that they are very easy to use off-the-shelf, context-independent models represent a particularly promising approach to predicting ratings of complexity in context.

## References

- Marc Brysbaert, Amy B. Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904-911.
- Marc Brysbaert, Paweł Mandera, Samantha F. McCormick, and Emmanuel Keuleers. 2019. Word prevalence norms for 62,000 English lemmas. *Behavior Research Methods*, 51(2): 467-479.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *Proceedings of the ICLR*. Pages 1-18.
- Council of Europe. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Press Syndicate of the University of Cambridge, Cambridge, United Kingdom.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the NAACL-HLT*. Association for Computational Linguistics, pages 4171-4186.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. arXiv preprint arXiv:2006.03654.
- Paul Hoffman, Matthew A. Lambon Ralph, and Timothy T. Rogers. 2013. Semantic diversity: A measure of semantic ambiguity based on variability in the contextual usage of words. *Behavior Research Methods*, 45(3):718-730.
- Geoff Hollis, Chris Westbury, and Lianne Lefsrud. 2017. Extrapolating human judgments from skip-gram vector representations of word meaning. *Quarterly Journal of Experimental Psychology*, 70(8):1603-1619.
- Brendan T. Johns, Randall K. Jamieson, and Michael N. Jones. 2020. The continued importance of theory: Lessons from big data approaches to language and cognition. In *Big data methods for psychological research: New horizons and challenges*. APA, pages 277-295.
- Emmanuel Keuleers, Paula Lacey, Kathleen Rastle, and Marc Brysbaert. 2012. The British Lexicon Project: Lexical decision data for 28,730 monosyllabic and disyllabic English words. *Behavior Research Methods*, 44(1):287-304.
- Victor Kuperman, Hans Stadthagen-Gonzalez and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 English words. *Behavior Research Methods*, 44(4):978-990.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *Proceedings of the ICLR*. Pages 1-17.
- Yuri Lin, Jean-Baptiste Michel, Erez L. Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the Google Books Ngram Corpus. In *Proceedings of the 50th Annual Meeting of the ACL*. Association for Computational Linguistics, pages 169-174.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv preprint:1907.11692.
- Dermot Lynott, Louise Connell, Marc Brysbaert, James Brand, and James Carney. 2019. The Lancaster Sensorimotor Norms: Multidimensional measures of perceptual and action strength for 40,000 English words. *Behavior Research Methods*, 52:1-21.
- Mounica Maddela and Wei Xu. (2018). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 3749-3760.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the ICLR*. Pages 1-12.

- Saif M. Mohammad. 2018. Obtaining reliable human ratings of valence, arousal, and dominance for 20,000 English words. In *Proceedings of the ACL - Long Papers*. Association for Computational Linguistics, pages 174-184.
- Gustavo H. Paetzold and Lucia Specia. 2016. Inferring psycholinguistic properties of words. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 435-440.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pages 1532-1543.
- Armand Rotaru. 2020. ANDI @ CONCRETEXT: Predicting concreteness in context for English and Italian using distributional models and behavioural norms. In *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020)*. Online. CEUR.org.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A New Corpus for Lexical Complexity Prediction from Likert Scale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*. European Language Resources Association, pages 57-62.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021. Predicting Lexical Complexity in English Texts. arXiv preprint arXiv: 2102.08773.
- Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. ConceptNet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI*. AAAI Press, pages 4444-4451.
- Akira Utsumi. 2020. Exploring What Is Encoded in Distributional Word Vectors: A Neurobiologically Motivated Analysis. *Cognitive Science*, 44(6):e12844.
- Walter J.B. Van Heuven, Pawel Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2014. SUBTLEX-UK: A new and improved word frequency database for British English. *Quarterly Journal of Experimental Psychology*, 67(6):1176-1190.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the EMNLP Workshop BlackboxNLP*. Association for Computational Linguistics, pages 353-355.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, pages 38-45.



# JUST-BLUE at SemEval-2021 Task 1: Predicting Lexical Complexity using BERT and RoBERTa Pre-trained Language Models

Tuqa Bani Yaseen Qusai Ismail Sarah Al-Omari Eslam Al-Sobh Malak Abdullah

Jordan University of Science and Technology

Irbid Jordan

tmbaniyaseen19, qfismail20, ssalomari18, esalsobh19 @ cit.just.edu.jo

mabdullah@just.edu.jo

## Abstract

Predicting the complexity level of a word or a phrase is considered a challenging task. It is even recognized as a crucial step in numerous NLP applications, such as text rearrangements and text simplification. Early research treated the task as a binary classification task, where the systems anticipated the existence of a word's complexity (complex versus uncomplicated). Other studies had been designed to assess the level of word complexity using regression models or multi-labeling classification models. Deep learning models show a significant improvement over machine learning models with the rise of transfer learning and pre-trained language models. This paper presents our approach that won the first rank in the SemEval-task1 (sub task1). We have calculated the degree of word complexity from 0-1 within a text. We have been ranked first place in the competition using the pre-trained language models BERT and RoBERTa, with a Pearson correlation score of 0.788.

keywords: Neuro-linguistic programming (NLP), Lexical Complexity Prediction(LCP), Deep Learning, RoBERTa, BERT.

## 1 Introduction

Lexical complexity plays a significant role in the readability level and comprehension. The precise anticipation of lexical complexity can help systems direct the user to an acceptable simple text accurately or modify the text to be more fluid (Brothers and Traxler, 2016). Predicting the complexity of words is a subjective and challenging problem, while it is conjectural, too. Yet, mapping words into their complexity is an essential task to understand natural language. Numerous components can influence the prediction of lexical complexity. Several approaches were proposed to solve or mitigate this type of study using Machine and Deep learn-

ing methods (Sengupta et al., 2020; Gooding and Kochmar, 2019; Bahja, 2020).

This paper describes the JUST-BLUE team's model that participated in the SemEval 2021-task1, Lexical Complexity Prediction (LCP) (Shardlow et al., 2021). The task provides participants with an augmented version of CompLex, a multi-domain English dataset with sentences annotated using a 5-point Likert scale (1-5) (from very easy to very difficult) (Shardlow et al., 2020). The task is to predict the complexity value of words in context. It is worth mentioning that our model, JUST-BLUE, has been ranked first in this task. We have used the pre-trained language models, BERT and RoBERTa Which have proven their effectiveness in this area (Liu et al., 2019), along with the ensemble method (weighted averaging) to achieve the highest Pearson correlation score of 0.788.

The rest of this paper is organized as follows: Section 2 sheds light on related work. Section 3 describes the methodology proposed in this research. Section 4 discusses the experimentation setup and evaluation results. Whereas Section 5 concludes this research.

## 2 Related work

One of the most prominent challenges in the current era is the prediction of lexical complexity. Prediction of the word complexity in machine learning can be binary; the word is complex or not complex. It also can be a non-binary prediction, as a probabilistic prediction with the measurement of complexity within a particular scale (0.6 the probability that the word is complex). SemEval 2016 introduced the first shared task of predicting word complexity with a mission limited to the word orders being complex or non-complex (binary prediction) (Paetzold and Specia, 2016). Decision Tree classifiers achieved the best results (Zampieri

et al., 2017). It has been noted that word length is a good indication of word complexity (De Hertog and Tack, 2018).

The authors in (Shardlow, 2013) discussed the importance of frequency and length of words. They used the Keras deep learning library to predict whether an English or Spanish word is complex or not. They used character embedding, word length, frequency count, word embedding, and psychological measures as features to predict complex words and achieved 0.872 as F1-score. The authors in (Yimam et al., 2018) worked on various languages, such as English, Spanish, French and German. They worked on two different methods for predicting complex words. The first method is to find if the word orders are either complex or simple. The second is to find the probability that the word is complex. The complex levels depended on the average of the annotators' answers. For example, if the number of annotators who expected the word to be complex is 6 out of 10, then the probability is 0.6. A claim stated that this annotating method is considered impractical since the probability of 0.5 cannot be considered complex or not complex. So the authors in (Shardlow et al., 2020) suggested a Likert scale with 5-point. The authors asserted that this method is more accurate scale instead of calling the word complex and non-complex. We can divide the word into being very easy, easy, neutral, difficult, and very difficult. This scale is beneficial to our work.

The deep learning pre-trained language models, BERT and RoBERTa, are considered state-of-the-art for NLP. Teams in the previous shared tasks of SemEval 2020 had used these models to obtain the best results for different NLP tasks (Al-Khdour et al., 2020; Shatnawi et al., 2020; Jurkiewicz et al., 2020). Our approach experimented with these models using different hyperparameters and weighted averaging methods that lead to the best result in the competition for predicting lexical complexity.

### 3 Methodology

This section describes our approach methodology and goes as follows: First, we describe the task's dataset. Then, the preprocessing step. Finally, we describe the JUST-BLUE approach to predict the word's complexity.

#### 3.1 Data

The SemEval-task 1 competition has provided the contestants with three files (trial, train, and test data). The files contain several columns as follows:

- id: the identification number for each entry.
- corpus: the sources from which the words were being collected. It was extracted from three sources: the bible, biomedical, and The European Parliament.
- sentence: the set of words for which complexity needed to be measured.
- token: the single word in which complexity needed to be measured.
- complexity: the degree of complexity of the word, ranging from 0 to 1.

#### 3.2 Pre-Processing Step

First, we cleaned the data and removed all single and double quotations manually. This step helped to separate some of the merged rows. Next, we deleted any row where columns contain the NaN value because it will not be effective in the training process.

#### 3.3 JUST-BLUE Architecture

We have used the pre-trained language models, BERT and RoBERTa models. We have imported the BERT model using BERT-sklearn library as it includes SciBERT and BioBERT models for the scientific and biomedical fields. We also have used simple transformers; classification libraries to import the RoBERTa model. As we mentioned earlier, the goal of the task is to determine the complexity of the word. Knowing that the word's complexity changes slightly based on the complexity of the sentence, we have used both the token (word) and the sentence to predict the word's complexity. We have fed BERT and RoBERTa models with the 'token,' and the 'complexity' label to be trained. We have also inserted 'sentence' and 'complexity' columns to both models for training as a second strategy. The results have been combined using an ensembling voting method, Weighted Averaging. Our experiments show that the 80:20 ratio for weights can achieve the best results. The highest voting rate is for the "token" model (model 1) since we need to calculate the degree of complexity for a single word. On the other hand, the complexity of a

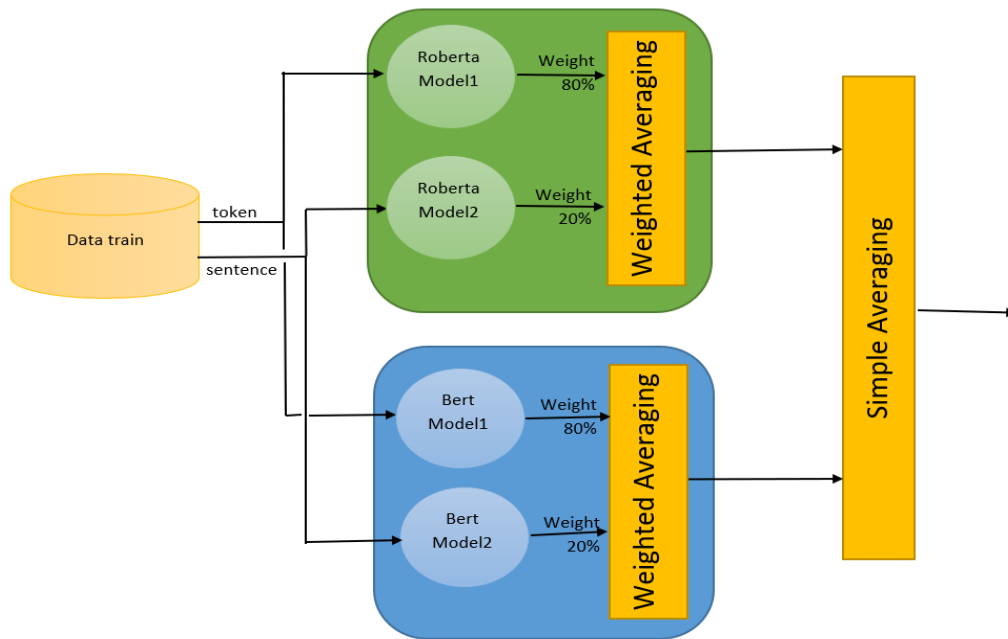


Figure 1: JUST-BLUE workflow

word is affected by the complexity of the sentence in which it is included. So, we gave a 20

The Simple Averaging method has been used as the ensembling technique to merge BERT and RoBERTa’s models’ results. Figure 1 illustrates the methodology used.

For more clarification, suppose we have the word ‘sea’ for which we want to calculate the complexity. The ‘sea’ word exists in this sentence ”and they entered into the boat, and were going over the sea to Capernaum.” First, we feed the word sea to model1 using RoBERTa. We also feed the sentence that contains the word sea to RoBERTa model2. Then, we combine the two results obtained using Weighted Averaging. Suppose that the RoBERTa model1 result is 0.01 (the word sea has a 0.01 complexity degree) and RoBERTa model2 is 0.13 ( the sentence has a 0.13 complexity degree). The resulted RoBERTa models is  $0.01 \times 80\% + 0.13 \times 20\%$ , which is equal to 0.034. We repeat these steps for BERT’s models. If the BERT model has a result of 0.052, then the final step is to calculate the average of the RoBERTa and BERT model. The complexity is  $(0.034 + 0.052)/2$ , equal to 0.043, as shown in Figure 2.

#### 4 Results and Discussion

We used Python version 3.6 on the Colab environment to execute our codes. We have experimented

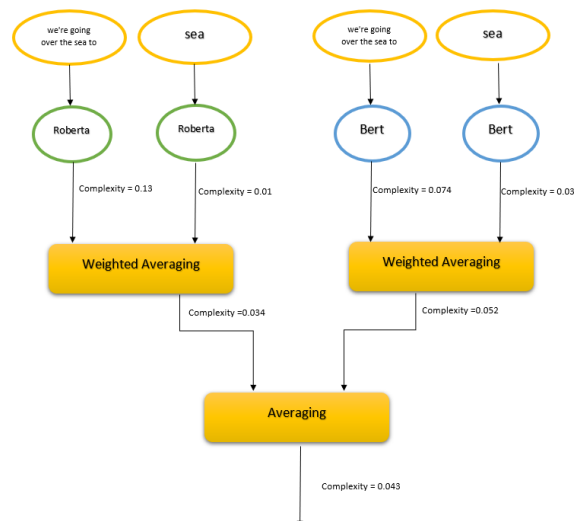


Figure 2: Example Description

with several models to determine which models are suitable for this task. We have experimented with BERT and RoBERTa pre-trained models. We also examined SVM and Random Forest machine learning models. Table 1 shows the results we have obtained throughout our experiments.

The challenging step was to find the best weights for the models that used tokens (single words) and sentences to get the best result (Table2). As we mentioned earlier, some words have a different complexity degree, depending on their location in the sentences. Therefore, it was necessary to insert

Table 1: Results of different models

Models	score
SVM	0.3472
Random Forest	0.4503
BERT	0.8199
RoBERTa	0.8268
BERT and RoBERTa	0.8190

Table 2: Different weights (tokens and sentences)

model		score
Token %	Sentence %	
90	10	0.8258
80	20	0.8268
70	30	0.8252

each of the words and sentences for the training to verify the best weight. Table 2 shows the best weight, which is 80% for words and 20% for sentences.

The next step was to explore BERT and RoBERTa’s best hyperparameters, such as learning rate, batch size, epochs, and max sequence length. Table 3 shows the description of these hyperparameters, and Table 4 shows example results of fine-tuning JUST-BLUE hyperparameters.

Finally, we thought of determining the effects of the base size and large size models of BERT and RoBERTa on the accuracy. It is shown by our experiments that the large sizes decreased the accuracy.

In the testing phase, we noticed that the words (tokens) in the file were new. Therefore, we decided to limit the number of arguments to avoid overfitting. We just changed ”num-train-epochs”=3 in BERT and RoBERTa’s model, but the other arguments had the default values. We have used three different models. The first was the BERT model, the second was the RoBERTa model, and the third was BERT and RoBERTa together as described in the Methodology Section. Table 5 shows the results we received from the different models we used.

JUST-BLUE approach achieved the best result using RoBERTa and BERT’s models with a Pearson correlation of 0.788 scores. We have also achieved the least Mean Absolute Error(MAE) with 0.0609. Our model is ranked first the LCP-sub task1 of a single word. The Spearman’s Rho (Rho) and R-squared (R2) scores are 0.7369 and 0.6172, respectively. The number of teams in the shared

task Lexical Complexity Prediction (LCP) was 54 teams. This shared task is considered a high level of CWI 2016 and CWI 2018 with a larger number of words from various sources.

## 5 Conclusion

Predicting the complexity of words is one of the most prominent tasks that the NLP research community strives to solve. It is worth noting that in 2016 and 2018, two tasks were issued to determine whether the word was complex or not. SemEval 2021 introduced task 1, Lexical Complexity Prediction (LCP) that aims to predict the word’s complexity from 0 to 1. This paper described the top-ranked team’s model, JUST-BLUE. The JUST-BLUE model obtained the highest Pearson Correlation score of 0.788 using the pre-trained language models BERT and RoBERTa. Our strategy depends on the ensembling methods, Simple and Weighted Averaging.

## References

- Nour Al-Khdour, Mutaz Bni Younes, Malak Abdullah, and AL-Smadi Mohammad. 2020. Justmasters at SemEval-2020 task 3: Multilingual deep learning model to predict the effect of context in word similarity. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 292–300.
- Mohammed Bahja. 2020. Natural language processing applications in business. In *E-Business*. IntechOpen.
- Trevor Brothers and Matthew J Traxler. 2016. Anticipating syntax during reading: Evidence from the boundary change paradigm. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(12):1894.
- Dirk De Hertog and Anaïs Tack. 2018. Deep learning architecture for complexword identification. In *Thirteenth Workshop of Innovative Use of NLP for Building Educational Applications*, pages 328–334. Association for Computational Linguistics (ACL); New Orleans, Louisiana.
- Sian Gooding and Ekaterina Kochmar. 2019. Complex word identification as a sequence labelling task. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at SemEval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*.

Table 3: Description of some argument

Argument	Description
learning rate	The learning rate for training
number of epochs	One forward pass and one backward pass of training examples
max sequence length	Maximum sequence length the model will support
train batch size	Determines the number of samples in each mini batch
early stopping metric	Stop training when early stopping metric doesn't improve
early stopping delta	Counts as a better checkpoint
evaluate during training steps	Performs evaluation at every specified number of steps

Table 4: Hyperparameter Fine-Tuning for JUST-BLUE model in the training phase

#	learning rate	number of epochs	max sequence length	train batch size	early stopping metric	early stopping delta	evaluate during training steps	score
1	4e-5	1	128	8	mcc	0	2000	0.796245
2	5e-5	1	64	16	eval_loss	0.01	1000	0.799285
3	4e-5	2	128	16	mcc	0.01	2000	0.80589
4	4e-5	3	128	8	eval_loss	0	2000	0.819013
5	4e-5	3	64	8	mcc	0.01	2000	0.827414
6	5e-5	3	32	16	eval_loss	0.02	1000	0.784574
7	4e-5	5	128	16	mcc	0	2000	0.815554
8	3e-5	5	64	8	eval_loss	0	500	0.818949
9	4e-5	4	128	8	mcc	0.01	1000	0.827172
10	3e-5	4	64	16	eval_loss	0.02	2000	0.82385

Table 5: Results of different models in the testing phase

Model	Score
RoBERTa	0.7772
BERT	0.7556
JUST-BLUE (RoBERTa and BERT)	0.7886

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Gustavo Paetzold and Lucia Specia. 2016. SemEval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Saptarshi Sengupta, Sanchita Basak, Pallabi Saikia, Sayak Paul, Vasilios Tsalavoutis, Frederick Atiah,

Vadlamani Ravi, and Alan Peters. 2020. A review of deep learning with special emphasis on architectures, applications and recent trends. *Knowledge-Based Systems*, 194:105596.

Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. Complex—a new corpus for lexical complexity prediction from likert scale data. *arXiv preprint arXiv:2003.07008*.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 task 1: Lexical complexity prediction. In *In Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Fara Shatnawi, Malak Abdullah, and Mahmoud Hammad. 2020. Mlengineer at SemEval-2020 task 7: BERT-flair based humor detection model (bfhumor). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1041–1048.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. *arXiv preprint arXiv:1710.04989*.

# BigGreen at SemEval-2021 Task 1: Lexical Complexity Prediction with Assembly Models

Aadil Islam, Weicheng Ma, Soroush Vosoughi

Department of Computer Science

Dartmouth College

{aadil.islam.21, weicheng.ma.gr, soroush.vosoughi}@dartmouth.edu

## Abstract

This paper describes a system submitted by team `BigGreen` to LCP 2021 for predicting the lexical complexity of English words in a given context. We assemble a feature engineering-based model with a deep neural network model founded on BERT. While BERT itself performs competitively, our feature engineering-based model helps in extreme cases, eg. separating instances of easy and neutral difficulty. Our handcrafted features comprise a breadth of lexical, semantic, syntactic, and novel phonological measures. Visualizations of BERT attention maps offer insight into potential features that Transformers models may learn when fine-tuned for lexical complexity prediction. Our ensembled predictions score reasonably well for the single word subtask, and we demonstrate how they can be harnessed to perform well on the multi word expression subtask too.

## 1 Introduction

Lexical simplification (LS) is the task of replacing difficult words in text with simpler alternatives. It is relevant in reading comprehension, where early studies have shown infrequent words lead to more time spent by a reader fixated on it, and that ambiguity in a word’s meaning adds to comprehension time (Rayner and Duffy, 1986). Complex word identification (CWI) is believed to be a fundamental step in the automation of lexical simplification (Shardlow, 2014). Early techniques for conducting CWI suffer from a lack of robustness, from simplifying all words to *then* study its efficacy (Devlin, 1998), to applying thresholds on features like word frequency (Zeng et al., 2005).

This year’s Lexical Complexity Prediction (LCP) shared task (Shardlow et al., 2021) forgoes the treatment of word difficulty as a binary classification task (Paetzold and Specia, 2016a; Yimam et al.,

2018) and instead measures degree of complexity on a continuous scale. This choice is intriguing as it mitigates a dilemma with previous approaches of having to treat words extremely close to a decision boundary (suppose a threshold deems a word’s difficulty) identically to those that are far away, ie. extremely easy or extremely difficult.

Teams are asked to submit predictions on unlabeled test sets for two subtasks: predicting on English single word and multi word expressions (MWEs). For each subtask, `BigGreen` presents a machine learning-based approach that fuses the predictions of a feature engineering-based regressor with those of a feature learning-based deep neural network model founded on BERT (Devlin et al., 2018). Our code is made available on GitHub.<sup>1</sup>

## 2 Related Work

Previous studies have looked at estimating the readability of a given text at the sentence-level. Mc Laughlin (1969) regresses the number of polysyllabic words in a given lesson against the mean score for students quizzed on said lesson, yielding the SMOG Readability Formula. Dale and Chall (1948) offer a list of 768 (later updated to 3,000) words familiar to grade-school students in reading, which they find correlates with passage difficulty. An issue with traditional readability metrics seems to be the loss of generality at the word-level.

Shardlow (2013) tries a brute force approach where a simplification algorithm is applied to each word of a given text, deeming a word complex only if it is simplified. However, this suffers from the assumption that a non-complex word does not require further simplification. They also try assigning a familiarity score to a word, and determining whether the word is complex or not by applying a threshold.

<sup>1</sup><https://github.com/Aadil101/BigGreen-at-LCP-2021>

Corpus	Subtask	Train	Trial	Test
Bible	Single Word	2574	143	283
	Multi Word	505	29	66
Biomed	Single Word	2576	135	289
	Multi Word	514	33	53
Europarl	Single Word	2512	143	345
	Multi Word	498	37	65

Table 1: LCP train, trial, and test sets.

We avoid thresholding our features in this study as we find it unnecessary, since raw familiarity scores can be used as features in regression-based tasks.

Results from CWI at SemEval-2016 (Zampieri et al., 2017) suggest vote ensembling predictions of the best performing models as an effective strategy, while several top-performing models (Paetzold and Specia, 2016b; Ronzano et al., 2016; Mukherjee et al., 2016) appear to use linguistic information beyond just word frequency. This inspires our use of ensemble techniques, and a foray into phonological features as a new point of research. Results from CWI at SemEval-2018 show feature engineering-based models outperforming deep learning-based counterparts, despite the latter having generally better performances since SemEval-2016.

### 3 Data

#### 3.1 CompLex Dataset

Shardlow et al. (2020) present CompLex, a novel dataset in which each target expression (a single word or two-token MWE) is assigned a continuous label denoting its lexical complexity. Each label lies in range 0-1, and represents the (normalized) average score given by employed crowd workers who record an expression’s difficulty on a 5-point Likert scale. We define a sample’s *class* as the bin to which its complexity label belongs, where bins are formed using the following mapping of complexity ranges:  $[0, 0.2) \rightarrow 1$ ,  $[0.2, 0.4) \rightarrow 2$ ,  $[0.4, 0.6) \rightarrow 3$ ,  $[0.6, 0.8) \rightarrow 4$ ,  $[0.8, 1] \rightarrow 5$ . Target expressions in CompLex have 0.395 average complexity and 0.115 standard deviation, reflecting an imbalance in favor of class 2 and 3 samples.

Each target expression is accompanied by the sentence it was extracted from, drawn from one of three corpora (Bible, Biomed, and Europarl). A summary of the train, trial,<sup>2</sup> and test set samples is

<sup>2</sup>In our study we avoid the trial set as we find it to be less representative of the training data, opting instead for training set cross-validation (stratified by corpus and complexity label).

provided in Table 1.

#### 3.2 External Datasets

We use four additional corpora to extract term frequency-based features from:

- **English Gigaword Fifth Edition** (Gigaword): this comprises articles from seven English newswires (Parker et al., 2011).
- **Google Books Ngrams, version 2** (GBND): this is used to count occurrences of phrases across a corpus of books, accessed via the PhraseFinder API (Trenkmann).
- **British National Corpus, version 3** (BNC): this is a collection of written and spoken English text (Consortium et al., 2007).
- **SUBTLEXus**: this consists of American English subtitles, offering a multitude of word frequency lists (Brysaert and New, 2009).

### 4 BigGreen System & Approaches

In this section, we overview features fed to our feature engineering-based model, as well as training techniques for the feature learning-based model. We describe our features in detail in Appendix A. Note that fitted models for the single word subtask are then harnessed for the MWE subtask.

#### 4.1 Feature Engineering-based Approach

##### 4.1.1 Feature Extraction

We aim to capture a breadth of information pertaining to the target word and its context. Most features follow heavily right-skewed distributions, prompting us to also consider the log-transformed version of each feature. For the MWE subtask, features are extracted independently for head and tail words.

##### 4.1.1.1 Lexical Features

These are features based on lexical information about the target word:

- **Word length**: length of the target word.
- **Number of syllables**: number of syllables in the target word, via the Syllables library.<sup>3</sup>
- **Is acronym**: whether the target word is a sequence of capital letters.

<sup>3</sup><https://github.com/prosegrinder/python-syllables>



#### 4.1.1.2 Semantic Features

These features capture the target word’s meaning:

- **WordNet features:** the number of hyponyms and hypernyms associated with the target word in WordNet (Fellbaum, 2010).
- **GloVe word embeddings:** we extract 300-dimension embeddings pre-trained on Wikipedia-2014 and Gigaword (Pennington et al., 2014) for each (lowercased) target word.
- **ELMo word embeddings:** we extract for each target word a 1024-dimension contextualized embedding pre-trained on the One Billion Word Benchmark (Peters et al., 2018).
- **GloVe context embeddings:** we obtain the average 300-dimension GloVe word embedding over each token in the given sentence.
- **InferSent context embeddings:** we obtain 4096-dimension InferSent embeddings (Conneau et al., 2017) for each sentence.

#### 4.1.1.3 Phonetic Features

These features compute the likelihood that soundable portions of the target word would arise in English language. We estimate ground truth transition probabilities between any two units (phonemes or characters) using Gigaword:

- **Phoneme transition probability:** we consider the min/max/mean/standard deviation over the set of transition probabilities for the target word’s phoneme bigrams.
- **Character transition probability:** analogous to that above, over *character* bigrams.

#### 4.1.1.4 Word Frequency & N-gram Features

These features are expressly included due to their expected importance as features (Zampieri et al., 2017). Gigaword is the main corpus from which we extract word frequency measures (for both lemmatized and unlemmatized versions of the target word), summed frequency of the target word’s byte pair encodings (BPEs), as well as summed frequencies of bigrams and trigrams. We complement these features with their IDF-based analogues. Lastly, we use the GBND, BNC, and SUBTLEXus corpora to extract secondary word frequency, bigram, and trigram measures.

#### 4.1.1.5 Syntactic Features

These are features that assess the syntactic structure of the target word’s context. We construct the constituency parse tree for each sentence using a Stanford CoreNLP pipeline (Manning et al., 2014).

- **Part of speech (POS):** tag is assigned using NLTK’s `pos_tag` method (Bird et al., 2009).
- **Depth of parse tree:** the parse tree’s height.
- **Depth of target word:** distance (in edges) between target word and parse tree’s root node.
- **Is proper:** whether the target word is a proper noun/adjective, detected using capitalization.

#### 4.1.2 Training

Prior to training, we Z-score standardize all features. For the single word subtask, we fit Linear, Lasso (Tibshirani, 1996), Elastic Net (Zou and Hastie, 2005), Support Vector Machine (Platt et al., 1999), K-Nearest Neighbors (Wikipedia, 2021), and XGBoost (Chen and Guestrin, 2016) regression models.

After identifying the best performing model by Pearson correlation, we seek to mitigate the imbalanced nature of the target variable, ie. multitude of class 1,2,3 and lack of class 4,5 samples: we devise a sister version of our top-performing model, fit upon a *reduced* training set. For the *reduced* set, we tune percentages removed from classes 1-3 by performing cross-validation on the full training set.

#### 4.2 Approach based on Feature Learning

Our handcrafted feature set relies heavily on target word-specific features. Beyond N-gram and syntactic features, it is a cursory analysis of the context surrounding the target word. We seek an alternative, automated approach using feature learning.

##### 4.2.1 Architecture

LSTM-based approaches have been used to model the contexts of target words in past works (Hartmann and Dos Santos, 2018; De Hertog and Tack, 2018). An issue with a single LSTM is its ability to read tokens of an input sentence sequentially only in a single direction (eg. left-to-right). It inspires us to try a Transformer-based approach (Vaswani et al., 2017), architectures that process sentences as a whole (instead of word-by-word) by applying *attention* mechanisms upon them. Attention weights

are useful as they can be interpreted as learned relationships between words. BERT (Devlin et al., 2018) is one such model used for a variety of natural language understanding (NLU) tasks.

Multi-Task Deep Neural Network (MT-DNN) proposed by Liu et al. (2019) offers state-of-the-art results for multiple NLU tasks by incorporating benefits of both multi-task learning and language model pre-training. We are able to initialize MT-DNN’s shared text encoding layers with a pre-trained BERT base model (cased), and fine-tune its later layers for 5 epochs, using a mean squared error loss function and default hyperparameters. Such hyperparameter settings are provided in Appendix B. Note that the model is fine-tuned on only the CompLex corpus.

#### 4.2.2 Input Layer

Data is fed to the model’s input layer in *Premise-AndOneHypothesis* format, premise and hypothesis being sentence and target word/MWE, respectively. The data is preprocessed by a BERT tokenizer, backed by Hugging Face (Wolf et al., 2020).

#### 4.2.3 Output Layer

Our model’s output layer produces the predicted lexical complexity for a given target word/MWE. Additionally, we extract *attention maps* across each of the model’s attention heads, for each test set sample. These will be assessed in Section 6.3.

#### 4.3 Ensembling

Our best performing feature engineering-based regression model yields two sets of predictions (from fitting on *full* and *reduced* training sets, respectively). We default to using the *full* predictions, then tune a threshold, where predictions higher than the threshold (likely of class 4,5 samples) are overwritten with the *reduced* predictions. We compute a weighted average ensemble of these predictions with those of our MT-DNN model to obtain a final set of predictions for the single word subtask.

For the MWE subtask, the fitted models from the previous subtask are harnessed to predict lexical complexities for the head and tail words. We then compute a weighted average ensemble of these predicted complexities *and* the predictions of an MT-DNN model trained on MWEs.

### 5 Results

We present the performances of BigGreen’s system on each subtask in Tables 2 and 3.

Model	Pearson	Rank
XGBoost <sub>full</sub>	0.7589	-
XGBoost <sub>reduced</sub>	0.7456	-
XGBoost <sub>full+reduced</sub>	0.7576	-
MT-DNN	0.7484	-
Ensemble (submission)	0.7749	8/54
Best competition results	0.7886	

Table 2: Test set results for single word subtask.

Model	Pearson	Rank
XGBoost <sub>full+red.</sub> (head)	0.7164	-
XGBoost <sub>full+red.</sub> (tail)	0.7188	-
MT-DNN	0.7890	-
Ensemble (submission)	0.7898	25/37
Ensemble (improved)	0.8290	*14/37
Best competition results	0.8612	

Table 3: MWE subtask test set results. (\*projection)

## 6 Analysis

### 6.1 Performance

For feature selection, we find success in selecting the top-300 features by mutual information and removing quasi-constant features. The pruned feature set is passed to wrapper/embedded methods and a variety of regressors for model comparison. We find an XGBoost regressor (with hyperparameters tuned via grid search) to excel consistently for the single word subtask. As shown in Table 2, we rank in the top 15% by Pearson correlation.

For the MWE subtask, performances are reported in Table 3. Note that our submitted predictions differ from post-competition predictions. We *previously* used a training procedure resembling that for the single word subtask: (1) filter methods for feature selection, (2) XGBoost for regression, (3) ensembling with MT-DNN. We had passed the entire MWE as input to our XGBoost and MT-DNN models. We hypothesize that the fewer number of training samples available for this subtask contributed to the prior procedure’s lackluster performance. This inspired us to incorporate the predictive capabilities of our fitted single word subtask models by applying them *independently* on the MWE’s constituent head and tail words. This gives us predicted complexities for the head and tail words each, which when ensembled with the predictions of our MT-DNN model (that, mind you, is trained on the *entire* MWE) yields superior results to those submitted to competition.

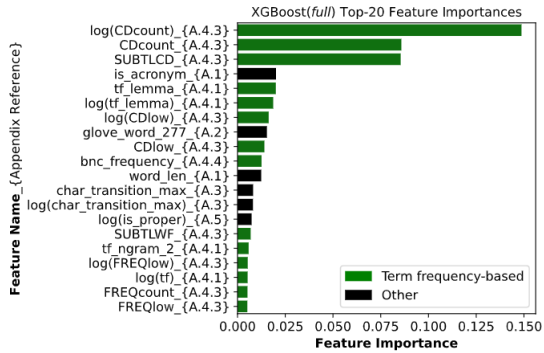


Figure 1: Feature importances for XGBoost<sub>full</sub>. Definitions of the features are shown in Appendix A.

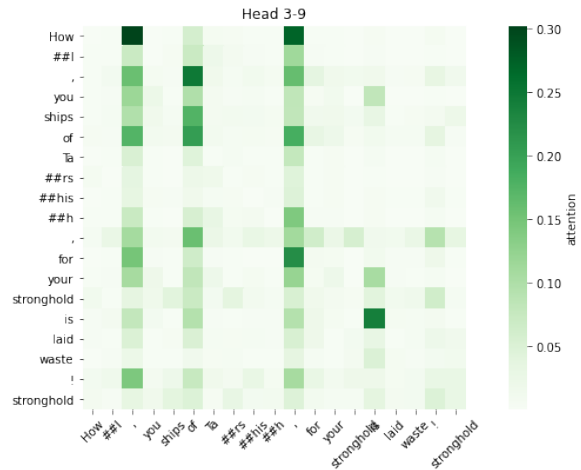


Figure 3: Head 3-9 attention map for a random sample.

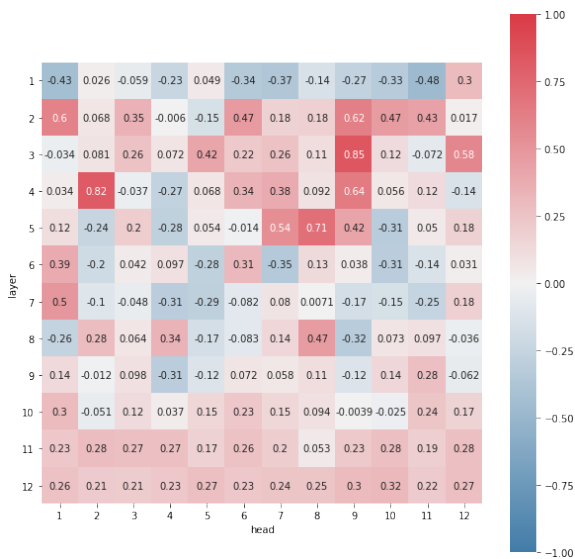


Figure 2: Attention head correlation between word frequency and total attention received by word, averaged across 100 random test set samples.

## 6.2 Feature Contribution

In total we consider 110 features, in addition to our multidimensional embedding-based features and log-transformed features. We inspect the estimated feature importance scores produced by the XGBoost<sub>full</sub> model to find that term frequency-based features (eg. unigrams, bigrams, trigrams) are of overwhelming importance (see Figure 1). This raises concern for whether the MT-DNN model too relies on term frequencies to make *its* predictions, and if not, the linguistic features it may have learned upon fine-tuning. Of the remaining features having non-zero feature importances, most appear to be dimensions of target word-based semantic features (ie. GloVe or ELMo embeddings).

## 6.3 BERT Attention

Attention maps have in previous works been assessed to demonstrate linguistic phenomena learned by a Transformer’s specialized attention heads (Voita et al., 2019; Clark et al., 2019). We extract attention maps from MT-DNN’s underlying fine-tuned BERT architecture. For each sample in the single word test set, we obtain an attention map from each of the BERT base model’s 144 attention heads (ie. 12 heads per 12 layers).

Based on the precedence given to term frequency features by the XGBoost<sub>full</sub> model, we hypothesize that for certain attention heads, the degree to which BPEs attend to one another varies relative to their word’s rarity in lexicon. This follows the findings of Voita et al. (2019), who identify heads in which lesser frequent tokens are attended to semi-uniformly by a majority of sentence tokens.

To test our hypothesis, we estimate for each attention head the Pearson correlation between word frequency and average attention given to each word in the context.<sup>4</sup> As illustrated in Figure 2, we find multiple attention heads appearing to specialize at directing attention towards the most or least frequent words (depending on sign of the correlation). Vertical stripe patterns like that in Figure 3 emerge as a result of attention originating from a spectrum of tokens. The findings seem to affirm the fundamental relevancy of word frequency to lexical complexity prediction, corroborating our intuition.

<sup>4</sup>We compute attention given to a *word* as the sum of attention given to its constituent BPEs. We use the GBND corpus to extract word frequencies, though any large corpora would suffice.

## 7 Conclusion

In this paper, we report inspirations for a system submitted by BigGreen to LCP SharedTask 2021, performing reasonably well for the single word subtask by adapting ensemble methods upon feature engineering and feature learning-based models. We see potential in future deep learning approaches, acknowledging the need for complementary word frequency-based handcrafted features for the time being. We surpass our submitted results for the MWE subtask, by utilizing the predictive capabilities of our single word subtask models.

Avenues for improvement include better data aggregation, as a relative lack of class 4,5 samples seems to hurt Pearson correlation across extremely complex samples. An approach may involve synthetic data generation using SMOGN (Branco et al., 2017). Shardlow et al. (2020) acknowledge a reader’s familiarity with a genre may affect perceived word complexity. However, the CompLex dataset lacks information on each annotator’s expertise or background, which may offer valuable new insights.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O’Reilly Media, Inc.
- Paula Branco, Luís Torgo, and Rita P Ribeiro. 2017. SMOGN: A pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications*, pages 36–50. PMLR.
- Marc Brysbaert and Boris New. 2009. Moving beyond Kučera and Francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for American English. *Behavior research methods*, 41(4):977–990.
- Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What Does BERT Look At? An Analysis of BERT’s Attention. *arXiv preprint arXiv:1906.04341*.
- Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.
- BNC Consortium et al. 2007. British national corpus. *Oxford Text Archive Core Collection*.
- Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.
- Dirk De Hertog and Anaïs Tack. 2018. Deep Learning Architecture for ComplexWord Identification. In *Thirteenth Workshop of Innovative Use of NLP for Building Educational Applications*, pages 328–334. Association for Computational Linguistics (ACL); New Orleans, Louisiana.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.
- Christiane Fellbaum. 2010. WordNet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.
- Nathan Hartmann and Leandro Borges Dos Santos. 2018. NILC at CWI 2018: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340.
- Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.
- Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.
- Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.
- G Harry Mc Laughlin. 1969. SMOG grading—a new readability formula. *Journal of reading*, 12(8):639–646.
- Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. JUNLP at SemEval-2016 Task 11: Identifying complex words in a sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 986–990.

- Gustavo Paetzold and Lucia Specia. 2016a. SemEval 2016 Task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.
- Gustavo Paetzold and Lucia Specia. 2016b. SV000GG at SemEval-2016 Task 11: Heavy gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974.
- R Parker, D Graff, J Kong, K Chen, and K Maeda. 2011. English Gigaword Fifth Edition LDC2011T07 (tech. rep.). Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.
- Keith Rayner and Susan A Duffy. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & cognition*, 14(3):191–201.
- Francesco Ronzano, Luis Espinosa Anke, Horacio Sagion, et al. 2016. TALN at SemEval-2016 Task 11: Modelling complex words by contextual, lexical and semantic features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1011–1016.
- Matthew Shardlow. 2013. A Comparison of Techniques to Automatically Identify Complex Words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.
- Matthew Shardlow. 2014. Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *LREC*, pages 1583–1590.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A New Corpus for Lexical Complexity Prediction from Likert Scale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.
- Martin Trenkmann. PhraseFinder – search millions of books for language use. <https://phrasefinder.io>. Accessed: 2021-02-08.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Senrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.
- Wikipedia. 2021. K-nearest neighbors algorithm — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=K-nearest%20neighbors%20algorithm&oldid=1008084290>. [Online; accessed 02-April-2021].
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.
- Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. *arXiv preprint arXiv:1710.04989*.
- Qing Zeng, Eunjung Kim, Jon Crowell, and Tony Tse. 2005. A text corpora-based estimation of the familiarity of health terminology. In *International Symposium on Biological and Medical Data Analysis*, pages 184–192. Springer.
- Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.

## A Feature Descriptions

Here, we describe in greater detail the various features that were experimented with for our feature engineering-based model. Note that while this discussion regards the single word subtask, for the MWE subtask we compute the same features but for each of the head and tail words, respectively.

### A.1 Lexical Features

`word_len`

- Character length of the target word.

`num_syllables`

- Number of syllables in the target word, via the Syllables library.

`is_acronym`

- Boolean for whether the target word is all capital letters.

### A.2 Semantic Features

`num_hyperyms`

- Number of hyperyms associated with the target word. The target word is initially disambiguated using NLTK's implementation of the Lesk algorithm for Word Sense Disambiguation (WSD) (Lesk, 1986), which finds the WordNet Synset with the highest number of overlapping words between the context and different definitions of each Synset.

`num_hyponyms`

- Number of hyponyms associated with the target word. Procedure for finding this is analogous to that for `num_hyperyms`.

`glove_word`

- 300-dimension embedding for each target word, pre-trained on Wikipedia-2014 and Gigaword. Target word is lowercased for ease.

`elmo_word`

- 1024-dimension embedding for each target word, pre-trained on the One Billion Word Benchmark corpus.

`glove_context`

- 300-dimension average of GloVe word embeddings (see `glove_word` above) for each word in the given context. Each word is lowercased for simplicity.

`inferred_embeddings`

- 4096-dimension embedding for the context.

### A.3 Phonetic Features

`char_transition_min`

- Minimum of the set of character transition probabilities for each character bigram in the target word. Ground truth character transition probabilities between any two English characters are estimated over Gigaword.

`char_transition_max`

- Maximum of the set described above.

`char_transition_mean`

- Mean of the set described above.

`char_transition_std`

- Standard deviation of the set described above.

`phoneme_transition_min`

- Minimum of the set of phoneme transition probabilities for each character bigram in the target word. Ground truth phoneme transition probabilities between any two phonemes are estimated over the Gigaword corpus. The phoneme set considered is that of the CMU Pronouncing Dictionary.<sup>5</sup>

`phoneme_transition_max`

- Maximum of the set described above.

`phoneme_transition_mean`

- Mean of the set described above.

`phoneme_transition_std`

- Standard deviation of the set described above.

### A.4 Word Frequency & N-gram Features

#### A.4.1 Gigaword-based

`tf`

- Target word term frequency. Note that all term frequency-based features are computed using Scikit-learn library's `CountVectorizer` (Pedregosa et al., 2011).

`tf_lemma`

- Term frequency of the lemmatized target word. Lemmatization is performed using NLTK's `WordNetLemmatizer`.

<sup>5</sup><http://speech.cs.cmu.edu/cgi-bin/cmudict>

tf\_summed\_bpe

- Sum of term frequencies of each BPE in the target word. BPE tokenization is performed using Hugging Face’s BERT Tokenizer.

tf\_ngram\_2

- Sum of the term frequencies of each bigram in the context containing the target word.

tf\_ngram\_3

- Sum of the term frequencies of each trigram in the context containing the target word.

tfidf

- Term frequency-inverse document frequency.

tfidf\_ngram\_2

- Sum of the term frequency-inverse document frequencies of each bigram in the context containing the target word.

tfidf\_ngram\_3

- Sum of the term frequency-inverse document frequencies of each trigram in the context containing the target word.

#### A.4.2 Google N-gram-based

google\_ngram\_1

- Term frequency of the target word.

google\_ngram\_2\_head

- Term frequency of leading bigram in the context containing the target word.

google\_ngram\_2\_tail

- Term frequency of trailing bigram in the context containing the target word.

google\_ngram\_2\_min

- Minimum of the set of term frequencies of bigrams in context containing the target word.

google\_ngram\_2\_max

- Maximum of the set described above.

google\_ngram\_2\_mean

- Average of the set described above.

google\_ngram\_2\_std

- Standard deviation of the set described above.

google\_ngram\_3\_head

- Term frequency of leading trigram in the context containing the target word.

google\_ngram\_3\_mid

- Term frequency of middle trigram in the context containing the target word.

google\_ngram\_3\_tail

- Term frequency of trailing trigram in the context containing the target word.

google\_ngram\_3\_min

- Minimum of set of term frequencies of trigrams in the context containing target word.

google\_ngram\_3\_max

- Maximum of the set described above.

google\_ngram\_3\_mean

- Average of the set described above.

google\_ngrams\_3\_std

- Standard deviation of the set described above.

#### A.4.3 SUBTLEXus-based

FREQcount

- Number of times target word appears in corpus.

CDcount

- Number of films in which target word appears.

FREQlow

- Number of times the lowercased target word appears in corpus.

CDlow

- Number of films in which the lowercased target word appears.

SUBTLWF

- Number of times the target word appears per million words.

SUBTLCD

- Percent of films in which target word appears.

#### A.4.4 BNC-based

bnc\_frequency: Target word term frequency.

#### A.5 Syntactic Features

parse\_tree\_depth

- Height of context's constituency parse tree. Parse trees are obtained using a Stanford CoreNLP pipeline.

token\_depth

- Depth of the target word with respect to root node of the context's constituency parse tree.

num\_words\_at\_depth

- Number of words at the depth of the target word (see `token_depth` above) in the context's constituency parse tree.

is\_proper

- Boolean for whether target word is a proper noun/adjective, based on capitalization.

POS\_{CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, LS, MD, NN, NNP, NNPS, NNS, PDT, POS, PRP, PRP\$, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP\$, WRB}

- Booleans indicating the target word's part-of-speech tag. Tags considered are those used in the Penn Treebank Project.<sup>6</sup> Tags are estimated using NLTK's `pos_tag` method.

#### A.6 Readability Features

automated\_readability\_index,  
avg\_character\_per\_word,  
avg\_letter\_per\_word,  
avg\_syllables\_per\_word,  
char\_count, coleman\_liau\_index,  
crawford, fernandez\_huerta,  
flesch\_kincaid\_grade,  
flesch\_reading\_ease,  
gutierrez\_polini,  
letter\_count, lexicon\_count,  
linsear\_write\_formula, lix,  
polysyllabcount, reading\_time,  
rix, syllable\_count,  
szigriszt\_pazos, SMOGIndex,  
DaleChallIndex

- Algorithms applied using Textstat library implementations, most being readability metrics.

<sup>6</sup>[https://www.ling.upenn.edu/courses/Fall\\_2003/ling001/penn\\_treebank\\_pos.html](https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

#### A.7 Other Features

ppl

- Perplexity metric, as defined by the Hugging Face library.<sup>7</sup> For each token in the context, we use a pre-trained GPT-2 model to estimate the log-likelihood of the token occurring *given its preceding tokens*. A sliding-window approach is used to handle the large number of tokens in a context. The log-likelihoods are averaged, and then exponentiated.

ppl\_aspect\_only

- Similar approach to that described above, where only log-likelihoods of tokens comprising the target word are averaged.

num\_OOV

- Number of words in the context that do not exist in the vocabulary of Gigaword.

corpus\_bible, corpus\_biomed,  
corpus\_europarl

- Booleans indicating the sample's domain.

## B Model Hyperparameters

Here we provide optimized hyperparameter settings that may help future developers with reproducing results, namely with training our models.

### B.1 XGBoost

Below are tuned parameters used for all of our XGBoost models. Parameters not listed are given default values as specified in documentation.<sup>8</sup>

```
colsample_bytree: 0.7  
learning_rate: 0.03  
max_depth: 5  
min_child_weight: 4  
n_estimators: 225  
nthread: 4  
objective: 'reg:linear'  
silent: 1  
subsample: 0.7
```

<sup>7</sup><https://huggingface.co/transformers/perplexity.html>

<sup>8</sup><https://xgboost.readthedocs.io/en/latest/>



## B.2 MT-DNN

MT-DNN uses `yaml` as its config file format. Below are the contents of our task config file:

```
data_format: PremiseAndOneHypothesis
enable_san: false
metric_meta:
- Pearson
- Spearman
n_class: 1
loss: MseCriterion
kd_loss: MseCriterion
adv_loss: MseCriterion
task_type: Regression
```

## B.3 Ensemble

Threshold above which a sample is assigned its *reduced* prediction (ie. `XGBoostreduced` prediction) instead of its *full* prediction (ie. `XGBoostfull` prediction): 0.59. Note that this threshold is used to compute our `XGBoostfull+reduced` prediction.

Weighted average ensemble (single word subtask):

- Weight for `XGBoostfull+reduced` prediction: 0.5
- Weight for MT-DNN prediction: 0.5

Weighted average ensemble (MWE subtask):

- Weight for `XGBoostfull+reduced(head)`: 0.28
- Weight for `XGBoostfull+reduced(tail)`: 0.17
- Weight for MT-DNN prediction: 0.55

# cs60075\_team2 at SemEval-2021 Task 1 : Lexical Complexity Prediction using Transformer-based Language Models pre-trained on various text corpora

Abhilash Nandy    Sayantan Adak    Tanurima Halder    Sai Mahesh Pokala  
{nandyabhilash, sayantanadak.skni, haldertanurima, pokalasaimahesh}@gmail.com  
Indian Institute of Technology, Kharagpur, India

## Abstract

This paper describes the performance of the team cs60075\_team2 at SemEval 2021 Task 1 - Lexical Complexity Prediction. The main contribution of this paper is to fine-tune transformer-based language models pre-trained on several text corpora, some being general (E.g., Wikipedia, BooksCorpus), some being the corpora from which the CompLex Dataset was extracted, and others being from other specific domains such as Finance, Law, etc. We perform ablation studies on selecting the transformer models and how their individual complexity scores are aggregated to get the resulting complexity scores. Our method<sup>1</sup> achieves a best Pearson Correlation of 0.784 in sub-task 1 (single word) and 0.836 in sub-task 2 (multiple word expressions).

## 1 Introduction

Complex words hinder the readability of a text, as discussed in (William, 2004). To mitigate this problem, there is a necessity of lexical simplification (Leroy et al., 2013), and predicting the complexity of words is an integral part of this process.

Language Models learn the probability of co-occurrence of words in a corpus. They have been used for various sentence completion and text-based classification tasks. The first language models were n-gram Markov Models (Rabiner and Juang, 1986), which performed well for tasks that did not require very long-range dependencies. Then came RNNs (Cho et al., 2014a), LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014b), which were able to understand longer contexts, but struggled with long paragraphs due to the vanishing gradient problem. Transformers (Vaswani et al., 2017) were a task-agnostic solution that performed better due

<sup>1</sup>The code is available at <https://github.com/abhilnandy2/CS60075-Team-2-Task-1>

to the presence of Attention Layers between hidden layers of the neural network, which helped the layers of the neural network to look at the entire input at once. Transformers can perform very well on a broad suite of tasks by fine-tuning on a small number of task-specific samples. The intuition behind using such transformer-based language models for Lexical Complexity Prediction (LCP) was - transformer models pre-trained on different corpora would mimic annotators (of the CompLex Dataset (com)) having different backgrounds. Since the final score is an aggregation of the annotation scores given by annotators, we aggregate the various scores that are given as outputs by the transformer-based models fine-tuned on the CompLex Dataset.

The rest of the paper is organized as follows. Section 2 gives an overview of our solution approach, Section 3 talks about the corpora used for pre-training and the dataset used for fine-tuning for the LCP task, Section 4 discusses the experimental settings, baselines used, and a comparison and analysis of the results and Section 5 gives a conclusion.

## 2 Solution Overview

### 2.1 Model Architecture

We use several transformer models. The general block diagram of such a model is shown in Fig. 1. The input to a model is the tokenized form of a sentence, and the tokenized form of the word/multi-word expression whose complexity score is to be predicted (separated by special tokens), and the target output is the complexity score. Each model consists of a transformer encoder, having the architecture of either BERT (Devlin et al., 2018) or RoBERTa (Liu et al., 2019), followed by a linear layer and a sigmoid activation layer so that the output is squashed in the range (0, 1). Sigmoid Ac-

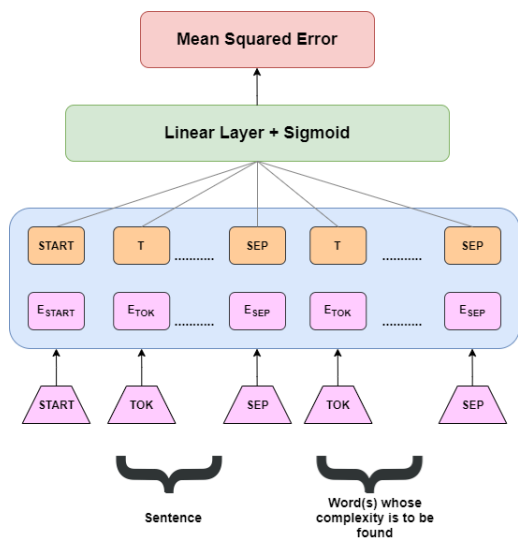


Figure 1: General Block Diagram of the Transformer

tivation Function is applied, as the target complexity score is a value between 0 and 1. To compute loss for backpropagation, the mean squared error loss function is used, as the problem is, as such, a regression problem.

## 2.2 Pre-training the transformer on text corpora

In order to initialize the weights and the embeddings of the transformer encoder, it is pre-trained on large text corpora so that it has syntactic, lexical and semantic knowledge before fine-tuning on the task-specific data. This is done for two reasons - (1) To increase the rate of convergence towards the lexical complexity prediction task (2) To mimic an annotator from a particular background.

In order to pre-train a transformer, specific pre-training tasks are performed. If the transformer being used is RoBERTa, Masked Language Modelling (MLM) is performed, where 15% of all the tokens are randomly replaced by a  $\langle MASK \rangle$  token. Such a masked sentence is provided as input to the transformer language model, and a Softmax Layer activation Function is applied for the output corresponding to the masked token to find out the probabilities of various tokens in the vocabulary being in the place of the  $\langle MASK \rangle$  in the original, unmasked sentence. The target is the actual token that was masked. A cross-entropy loss function is used to calculate the loss that is backpropagated. In the case of the BERT Transformer, in addition to the MLM pre-training task, Next Sentence Prediction (NSP) Task is also performed. Two sentences are taken from the corpus, where either one sen-

tence follows the other, or the two sentences are far apart. The output is either 1 corresponding to the sentences being adjacent to each other, and 0 being the case when they are far apart. Both the cases have the same number of samples while training. The output corresponding to the *START* (here,  $\langle CLS \rangle$ ) token is passed through a linear layer to get a 2x1 shaped vector, which is then followed by a Softmax Layer, thus giving probabilities of whether the second sentence comes after the first one or not.

## 3 Data

### 3.1 Data used for pre-training

Since we require several transformer language models pre-trained on a wide variety of corpora, we make it a point that we have transformers pre-trained on text corpora from which the CompLex Dataset has been extracted. These corpora are - (1) World English Bible Translation (Christodouloupoulos and Steedman, 2015) (We used the data found in this link <sup>2</sup>) (2) English part of the European Parliament Proceedings from europarl (Koehn, 2005) (3) CRAFT corpus (Bada et al., 2012) of bio-medical domain. We pre-train three RoBERTa language models on these three corpora (initialized by weights from (Liu et al., 2019)) using the MLM pre-training task.

### 3.2 Data used for fine-tuning

For fine-tuning, we do not use any external data other than the datasets that have been provided for both the sub-tasks <sup>3</sup>.

## 4 Experiments and Results

### 4.1 Transformer Language Models used

We use the predictions from 9 transformer-based language models, 4 of which have a RoBERTa encoder, and the other 5 have a BERT-based encoder. 2 models are pre-trained on general domain corpora like Wikipedia and BooksCorpus, 2 models on biomedical and clinical data, 2 models on Europarl data, 1 on Bible, 1 on Financial data, and 1 on scientific papers. Also, 6 of the pre-trained transformer models were publicly available in the HuggingFace Models Catalog <sup>4</sup>, while the other 3 were pre-trained by us on the three datasets from

<sup>2</sup><https://www.kaggle.com/oswinrh/bible>

<sup>3</sup><https://github.com/MMU-TDMLab/CompLex>

<sup>4</sup><https://huggingface.co/models>

APPROACH	Single word		MWE	
	PC	MSE	PC	MSE
xgb-A	0.718	0.0078	0.762	0.0103
xgb-B	0.741	0.0073	0.815	0.0083
xgb-C	0.744	0.0072	0.817	0.0082*
BERT-BASE-UNCASED	0.765	0.007*	0.791	0.009
BIBLE+EUROPARL+BIOMED (AVG.)	0.753	0.0075	0.798	0.0096
BIBLE+EUROPARL+BIOMED (MAX.)	0.751	0.0076	0.788	0.0092
BEST COMBINATION (AVG.)	<b>0.784</b>	<b>0.0066</b>	<b>0.836</b>	<b>0.0078</b>
BEST COMBINATION (MAX.)	0.774*	0.0071	0.819*	0.0091

Table 1: Comparing the Pearson Correlation (PC) and Mean Squared Error (MSE) of our methods and the baselines (The entries in **bold** are the best performing according to the respective column’s metrics, while the ones with a \* are the next best ones.)

which CompLex Dataset is extracted, as mentioned in Section 3.1.

## 4.2 Training, validation and Test Sets

For each sub-task, the training and the test sets are the same as those provided for the competition. The trial data given for each sub-task is taken to be the validation data.

## 4.3 Hyperparameters

For pre-training, the RoBERTa transformer language model, a batch size of 16 is used and is trained up to 1 epoch. The rest of the parameters are the same as in (Liu et al., 2019).

When fine-tuning, irrespective of whether the model has a RoBERTa or a BERT Transformer encoder, the input sequence length is set to 256, with padding or truncation, as is the case. A learning rate of  $2 \times 10^{-5}$  is used with a batch size of 32, and a Weighted Adam Optimizer is used. The network is fine-tuned for 4 epochs. The Pearson Correlation on the validation data is calculated for every epoch, and the checkpoint giving the best Pearson Correlation is regarded as the best checkpoint, which would later be used for predicting outputs on the test data.

## 4.4 Methods of Aggregation used

In order to aggregate the complexity scores of a particular combination of models, we use the following two strategies - sample-wise average and sample-wise maximum across all transformer models. We then do the same across all permutations,

see which combination gives the best test results and report it as the final result.

## 4.5 Baselines

We use XGBoost (Chen and Guestrin, 2016) to perform a boosting-based regression model, with an objective of squared error and other default parameters and hyperparameters over a set of features. The different baselines use different feature sets, which are as follows -

1. **xgb-A** - word length (sum of word lengths in the case of MWE), number of syllables and word frequency (from various text sources) (Speer et al., 2018) (average of word frequencies in the case of MWE) of the word/expression whose complexity is to be found, and the type of corpus of the sentence (either Bible, Europarl, or Biomedical).
2. **xgb-B** - Concatenation of features of xgb-A and the 50 and 100-dimensional GloVe (Pennington et al., 2014) word vectors of the word/expression whose complexity is to be found. For the expression, the sum of the GloVe Vectors of the individual words would be taken.
3. **xgb-C** - Concatenation of features of xgb-B and the probabilities of the word/expression whose complexity is to be found given the sentence with that word/expression that is masked, where the probabilities are predicted by different transformer-based masked language models pre-trained on different corpora.

**Note:** The probability of the word/token given the masked sentence is approximated as the product of the probabilities of predicting each token, given other tokens of the sentence are masked. E.g., Given a sentence  $S$  - "I just love mowing the lawn with a lawn mower." Let's say one is required to find out the complexity of the expression - "lawn mower". First, 'lawn' is masked in  $S$ , and the probability to predict 'lawn' using the transformer model  $M$  is found, denoted by  $P1$ . Similarly, 'mower' is masked in  $S$ , and the probability to predict 'mower' using  $M$  is found, denoted by  $P2$ . Resultant feature value =  $P1 * P2$

#### 4.6 Results and Discussion

Table 1 compares the Pearson Correlations (higher the better) and mean squared errors (lower the better) of our best (according to Pearson Correlation) aggregate results (for both average as well as maximum aggregation), some ablations, and the baselines for both the sub-tasks.

Based on the results, we can infer the following

1. **xgb-B** performs better than **xgb-A**, suggesting that, GloVe Word Vector features perform a vital role in complexity prediction, as they contain some contextual information regarding the word.
2. **xgb-C** performs the best among the baselines, as it also considers the probabilities of predicting the masked tokens whose complexity is found, adding to the contextual information.
3. Fine-tuning **BERT-BASE-UNCASED** transformer model for the LCP task performs better than the best baseline in case of sub-task 1, which could be attributed to the reason that, fine-tuning attention-based transformer models captures even more contextual information than the baselines.
4. Fine-tuning and aggregating RoBERTa Transformer models pre-trained on the three corpora from which the CompLex Dataset was extracted (**BIBLE+EUROPARL+BIOMED**), still gives better results than the baselines (except for **xgb-B** and **xgb-C** in case of sub-task 2), but performs inferior as compared to **BERT-BASE-UNCASED** model for single

word sub-task, while performing almost similar in case of Multi-Word Expressions sub-task. Also, the average aggregation performs better than the maximum aggregation.

5. The combination of transformer models that gives the best results upon aggregation (**BEST COMBINATION**), consists of 3-4 different transformer models fine-tuned on the dataset, suggesting that, **transformer models pre-trained on domains related as well as unrelated to the dataset (such as Financial Data, Legal Data), are able to best mimic annotators coming from various backgrounds**. Even in this case, average aggregation performs better than maximum aggregation.
6. If we consider the evaluation metrics of Pearson Correlation (PC) and Mean Squared Error (MSE), it can be seen (especially in the single-word sub-task) that they are negatively correlated, as is expected.

## 5 Conclusion

We show that aggregating the results of various fine-tuned transformer models pre-trained on various corpora from different domains gives high Pearson Correlation and low mean squared errors compared to individual transformers and regression models using attributes such as hand-crafted features, word embeddings, transformer-based language model prediction probabilities, etc. This shows that transformer-based language models, each pre-trained on a different text corpus, can better imitate annotators of the dataset, who come from diverse backgrounds and prior knowledge.

## References

- Complex: A new corpus for lexical complexity prediction from likert scale data.
- Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A Baumgartner, K Bretonnel Cohen, Karin Verspoor, Judith A Blake, et al. 2012. Concept annotation in the craft corpus. *BMC bioinformatics*, 13(1):1–20.
- Tianqi Chen and Carlos Guestrin. 2016. **XGBoost: A scalable tree boosting system**. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM.

- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014a. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language resources and evaluation*, 49(2):375–395.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86. Citeseer.
- Gondy Leroy, David Kauchak, and Obay Mouradi. 2013. A user-study measuring the effects of lexical simplification and coherence enhancement on perceived and actual text difficulty. *International journal of medical informatics*, 82(8):717–730.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.
- Lawrence Rabiner and Biinghwang Juang. 1986. An introduction to hidden markov models. *iee assp magazine*, 3(1):4–16.
- Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. [Luminosinsight/wordfreq: v2.2](#).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- HD William. 2004. The principles of readability. eric. *Online Submission*.

# C3SL at SemEval-2021 Task 1: Predicting Lexical Complexity of Words in Specific Contexts with Sentence Embeddings

**Raul Gomes Pimentel de Almeida**

C3SL Labs  
Universidade Federal do Paraná  
Curitiba - Brazil  
rgpa18@inf.ufpr.br

**Hegler Tissot**

Cognitive Computation Group  
University of Pennsylvania  
Philadelphia - USA  
hegler@seas.upenn.edu

**Marcos Didonet Del Fabro**

C3SL Labs  
Universidade Federal do Paraná  
Curitiba - Brazil  
marcos.ddf@inf.ufpr.br

## Abstract

We present our approach to predicting lexical complexity of words in specific contexts, as entered LCP Shared Task 1 at SemEval 2021. The approach consists of separating sentences into smaller chunks, embedding them with Sent2Vec, and reducing the embeddings into a simpler vector used as input to a neural network, the latter for predicting the complexity of words and expressions. Results show that the pre-trained sentence embeddings are not able to capture lexical complexity from the language when applied in cross-domain applications.

## 1 Introduction

Lexical complexity plays a crucial role in reading comprehension. Predicting lexical complexity of words within sentences in specific textual context can enable systems to better perform certain NLP tasks, such as simplifying texts, and favoring less fortunate readers in a giving target language. The SemEval 2021 proposes a Lexical Complexity Prediction (LCP) shared task (Task 1) (Shardlow et al., 2021) based on a new annotated English dataset with a Likert scale (Shardlow et al., 2020).

Word embeddings (Mikolov et al., 2013; Devlin et al., 2019) are very important resources that support several NLP tasks by providing a semantic latent representation that heuristically captures relationships in language that are very difficult to observe otherwise. Furthermore, such semantic representation can be used to embed language structures other than just words.

Sent2Vec (Pagliardini et al., 2018) is an unsupervised model designed to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training composition and the embedding vectors themselves. Sent2Vec has been used in several NLP tasks, such as analysing semantic properties of sentences (Zhu et al., 2018),

classification of sentences in the biomedical domain (Agibetov et al., 2018), automatic detection of incoherent speech (Iter et al., 2018), and measuring sentence similarity (Quan et al., 2019),

In this work, we aim to test the ability of Sent2Vec to detect the complexity of English words and expressions in specific contexts. We evaluate in what extent the semantic information captured when learning the embedding representation is able to incorporate word complexity.

Our approach uses pretrained Sent2Vec models and aims to validate in what extent such models are able to predict complexity of words. Results show strong evidence that pretrained sentence embeddings do not capture complexity features from language, specially when applied in cross-domain applications.

## 2 Method

The proposed shared task consists of determining the complexity of token words in the context of given input sentences (Lexical Complexity Prediction-LCP). Training input sentences annotated with a Likert scale corresponding to the complexity score of target words are given. Sub-task 1 focus on predicting the complexity score of single words. Meanwhile, Sub-task 2 targets multi-word expressions.

Our overall strategy consists on generating chunks of the sentence, embed those chunks with Sent2Vec and then reduce those embeddings into a simpler vector, which would then finally be used as the neural network's input.

Our approach uses pretrained Sent2Vec models to obtain embedding representation of multiple parts of the input sentence split by the target token words. For a given input sentence  $S$ , we obtain embeddings for: a)  $S$ : the full sentence; b)  $S_0$ : the amount of text from the beginning of the sentence

up to the target token(s) (except the latter); c)  $T$ : the target word token(s), and d)  $S_1$ : the amount of text from the target token(s) up to the end of the sentence (except the former). When target tokens are in the beginning or in the end of a sentence,  $S_0$  or  $S_1$  are represented by a zero-value vector.

We reduce the vector representation of each sentence constituent into distance-based arrays that are fed into a neural network estimator. The reduction itself consisted of simplifying the text (using Spacy) and then feeding different chunks into Sent2Vec. Finally, an array of the distances between the chunks' embeddings to the token word's embeddings was used as input to a neural network coming from the Scikit-Learn package.

We use the context given by the two splits of the input sentence ( $S_0$  or  $S_1$ ) to measure the complexity of the token word(s) ( $T$ ). Thus, our pipeline consists of four major steps: *text preprocessing*, *chunking*, *context embedding* and *estimator training*. The chosen estimator was a Multi-Layer Perceptron (MLP).

Figure 1 illustrates this pipeline for an example sentence and token word.

## 2.1 Text Preprocessing

We use a simple preprocessing step based on off-the-shelf components from Spacy Python package (Honnibal et al., 2020)<sup>1</sup> that apply the following filters in the raw input sentence:

- Turning it into only lowercase characters;
- Removing all punctuation;
- Removing all words recognized by Spacy as stop words (unless the stop word was part of the target token  $T$ ).

The reformatted sentences are then split into parts  $S$ ,  $S_0$ ,  $S_1$  and  $T$  to be fed into the next step of the pipeline.

## 2.2 Context Embedding

We embed the four elements of a sentence resulted from the previous preprocessing step using a pretrained Sent2Vec model (torontobooks\_unigrams model has been chosen).<sup>2</sup> As a setback, the token word was often embedded as a null vector (consisting only

<sup>1</sup><https://spacy.io>

<sup>2</sup><https://github.com/epfml/sent2vec#downloading-sent2vec-pre-trained-models>

of zeroes), as expected that such target words from specific domains would be off the pretrained vocabulary.

Instead of concatenating the four resulting embeddings into a single vector to be used as input by our estimator, we perform a reduction that aims to represent how close (distance) to its context a token word is. Thus, our estimator input comprises: a) three norm distances (NumPy's `linalg.norm` method (Harris et al., 2020)) between each sentence embedding ( $S$ ,  $S_0$ , and  $S_1$ ) and the token word's embedding ( $T$ ), and b) a boolean value in the set  $\{0, 1\}$  indicating whether the token word's embedding was a null vector.

## 2.3 Estimator Training

Finally, we use the obtained embedding reductions from the previous step to train a MLP from the Scikit-Learn Python package (`neural_network.MLPRegressor`) (Pedregosa et al., 2011). The MLP was instantiated with the `max_iter` and `random_state` arguments set to 500 and 1, respectively. The estimator was fitted to the input with its `fit` method, with the `X` and `y` parameters being vectors of the reduced embeddings ( $X$ ) and corresponding given token word complexities ( $y$ ).

Our estimator is designed to predict the complexity value for a given token word in context. However, token words can be a misuse of given information, since the complexity of a word varies with context. Our goal with reducing embeddings from fractions of the original sentence into a small vector that served as input to the estimator was similar to the Input Hypothesis, an approach to language learning that grows in popularity.

The Input Hypothesis, also referred to as “+1 method” and surveyed in Wang (2017), discusses a process for acquiring languages by being exposed to content that is slightly above the learner's current level. This is argued to work because the learner is then able to understand the whole sentence (i) and consequently annex a new word or construction (+1) to their vocabulary because of intelligible context.

Reducing embeddings from chunks of the original sentence into a vector of distances from the token word can be expressed as an attempt to symbolize the size (complexity) of this “+1” (the token word) - that is, the neural network input is an observation of how obtainable from its context a given



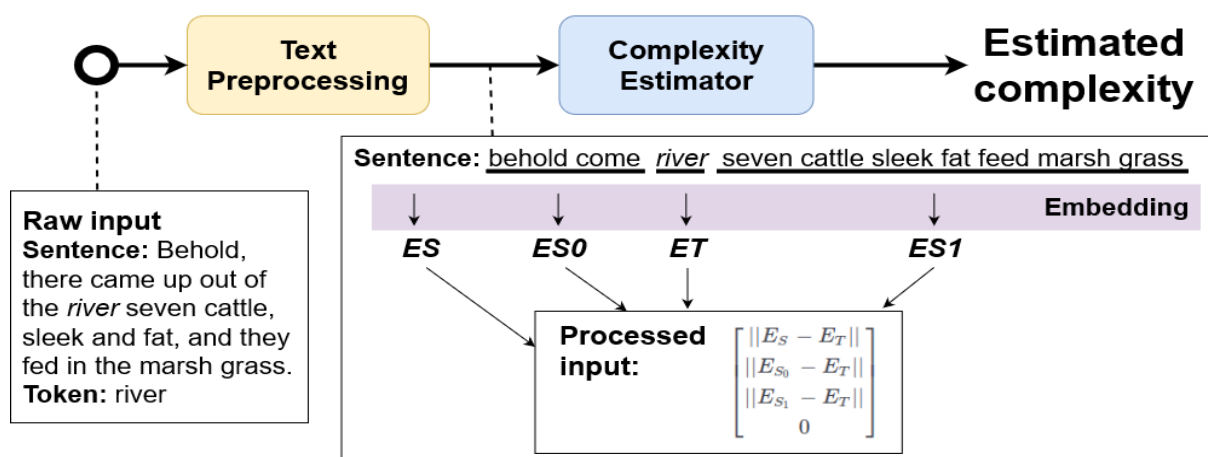


Figure 1: Illustrated pipeline for our approach

token word is.

We then validate the trained model using `model_selection.train_test_split` method (Scikit-Learn), setting `random_state` parameter set to 1. This method divides the dataset into training and testing subsets. With the MLP fitted with the training subset, the efficiency with which it would handle foreign inputs is measured with a call to its `score` function (using the testing sets as arguments). This function returns the coefficient of determination  $R^2$  of a prediction.

## 2.4 Development Environment

Our solution is made available in a Google Colab file<sup>3</sup>. In order to execute the code, it's necessary to duplicate the file and change the environment variables related to file access (the Sent2Vec models and input files). More information on this process can be found on the file itself.

Since the Google Colab platform permits by default the use of up to a little over 12.5GB RAM, the options of Sent2Vec pretrained models to use are limited: the model we used for this task has only 2GB.

## 3 Results

The task is divided between two subtasks: single- and multi-word tokens. Both have the same format: each line of the input has an ID, a sentence, a corpus to which it belongs, and a token. For training data, each line also includes a complexity value. The difference between the first and second subtasks is that the first limits a token to one single

<sup>3</sup>[https://colab.research.google.com/drive/1MCNfDzM-BW9Zopxs9qFFT8sDLb6FrAM\\_?usp=sharing](https://colab.research.google.com/drive/1MCNfDzM-BW9Zopxs9qFFT8sDLb6FrAM_?usp=sharing)

word, while the second doesn't.

Tables 1 and 2 present the final results of the competition for the two subtasks, respectively. Participant's performance is ranked in each subtask separately. For the submissions without a team name - that is, the submissions made by users not linked to teams -, the user name is available inside brackets. The tables also show each participant's scores for individual corpora. Our approach is identified as C3SL.

We believe the flaws in our approach can be explained in two-fold: a) pretrained embeddings are sensitive to the domain or context used during training, and the way semantic information is captured in a latent representation is not reflected the in the same way in cross-domain applications; and b) even if a latent representation of text would able to capture semantic complexity of certain expression in context, the same is not reflected by the norm or cosine distances between multiple chunks and the target expressions.

## 4 Related Work

Some of the previous work show that Sent2Vec is not the best alternative to consistently achieve high accuracy in subsequent NLP and NLU tasks based on embedding representation of sentences, except when training Sent2Vec with domain-specific corpora.

Miftahutdinov et al. (2019) attempt to use the Twitter unigram pretrained model from Sent2Vec in order to improve their approach when performing extraction of adverse drug reactions from Tweets. However, results show that utilizing Sent2Vec as tweet representations did not improve classification quality.

Table 1: Results for Subtask 1.

#	Team Name	Pearson	Spearman	MAE	MSE	R2
1	JUST BLUE	0.7886 (1)	0.7369 (2)	0.0609 (61)	0.0062 (60)	0.6172 (2)
2	DeepBlueAI	0.7882 (2)	0.7425 (1)	0.0610 (60)	0.0061 (61)	0.6210 (1)
3	Alejandro Mosquera	0.7790 (3)	0.7355 (5)	0.0619 (57)	0.0064 (59)	0.6062 (3)
57	C3SL	0.4598 (57)	0.3983 (58)	0.0866 (6)	0.0130 (6)	0.1989 (56)
61	RACAI	-0.0272 (61)	-0.0268 (61)	0.2777 (1)	0.1270 (1)	-6.8449 (61)

Table 2: Results for Subtask 2.

#	Team Name	Pearson	Spearman	MAE	MSE	R2
1	DeepBlueAI	0.8612 (1)	0.8526 (3)	0.0616 (38)	0.0063 (38)	0.7389 (1)
2	[rg_pa]	0.8575 (2)	0.8529 (2)	0.0672 (34)	0.0072 (34)	0.7035 (5)
3	[xiang_wen.tian]	0.8571 (3)	0.8548 (1)	0.0675 (33)	0.0072 (32)	0.7012 (7)
35	C3SL	0.3941 (35)	0.3675 (35)	0.1145 (4)	0.0206 (4)	0.1470 (35)
38	[glitterosu]	0.1860 (38)	0.1316 (38)	0.1332 (1)	0.0255 (1)	-0.0564 (38)

Cho et al. (2019) use SentVec embeddings to propose a language scheme that generates candidate utterances using paraphrasing and methods from semi-supervised learning.

An empirical study of sentence embeddings (Krasnowska-Kieraś and Wróblewska, 2019) aims to analyse in what extent linguistic information is retained in vector representations of sentences by comparing ten embeddings approaches. Results show that Sent2Vec was only able to outperform other approaches in only 1 out of 11 tasks (word classification task).

Lo et al. (2018) use Sent2Vec trained on the WMT18 news translation task parallel training data (Koehn et al., 2018) to calculate distance of sentence vectors aiming to improve their semantic textual similarity approach when filtering a noisy web crawled parallel corpus.

Iter et al. (2018) aim to automatically extract linguistic features for detecting symptoms of schizophrenia. They compare a number of sentence embeddings and show that, although Sent2Vec outperforms the mean vector sentence embedding (used as a baseline model within the experiments), all other models perform better when measuring coherence using concept overlap and ambiguous pronoun usage.

Zhu et al. (2018) evaluate semantic properties of sentence embeddings models in five tasks: negation detection, negation variants, clause relatedness, argument sensitivity, and fixed point reorder. Results show that Sent2Vec is only able to outperform other embedding models in the clause relatedness task,

which explores whether the similarity between a sentence and its embedded clause is higher than between a sentence and its negation. The resulting accuracy is in the 30-35% range.

## 5 Conclusions

We presented an approach for the LCP Shared Task 1 at SemEval 2021 for predicting the lexical complexities of words in context. We used pretrained Sent2Vec models using a vector formed from embedded chunks of sentences as input for a neural network to perform as the final complexity estimator. Implemented and executed on the Google Colab platform, our approach used little resources.

The results show that pretrained Sent2Vec models alone cannot capture a semantic representation that reflects a word’s or expression’s complexity within cross-domain contexts. As a way of extending our experiments, we plan to perform further analysis in subsequent classification tasks in the biomedical domain using reinforcement strategies to enrich the semantic information captured by embedding representation of sentences.

## References

- Asan Agibetov, Kathrin Blagec, Hong Xu, and Matthias Samwald. 2018. [Fast and scalable neural embedding models for biomedical sentence classification](#). *BMC Bioinformatics*, 19(1).
- Eunah Cho, He Xie, and William M. Campbell. 2019. [Paraphrase generation for semi-supervised learning](#)

- in NLU. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 45–54, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'io, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. [Array programming with NumPy](#). *Nature*, 585(7825):357–362.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Dan Iter, Jong Yoon, and Dan Jurafsky. 2018. [Automatic detection of incoherent speech for diagnosing schizophrenia](#). In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 136–146, New Orleans, LA. Association for Computational Linguistics.
- Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L. Forcada. 2018. [Findings of the WMT 2018 shared task on parallel corpus filtering](#). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.
- Katarzyna Krasnowska-Kieraś and Alina Wróblewska. 2019. [Empirical linguistic study of sentence embeddings](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5729–5739, Florence, Italy. Association for Computational Linguistics.
- Chi-kiu Lo, Michel Simard, Darlene Stewart, Samuel Larkin, Cyril Goutte, and Patrick Littell. 2018. [Accurate semantic textual similarity for cleaning noisy parallel corpora using semantic machine translation evaluation metric: The NRC supervised submissions to the parallel corpus filtering task](#). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 908–916, Belgium, Brussels. Association for Computational Linguistics.
- Zulfat Miftahutdinov, Ilseyar Alimova, and Elena Tutubalina. 2019. [KFU NLP team at SMM4H 2019 tasks: Want to extract adverse drugs reactions from tweets? BERT to the rescue](#). In *Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task*, pages 52–57, Florence, Italy. Association for Computational Linguistics.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. [Distributed representations of words and phrases and their compositionality](#). In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.
- Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. [Unsupervised learning of sentence embeddings using compositional n-gram features](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. [Scikit-learn: Machine learning in python](#). *Journal of machine learning research*, 12(Oct):2825–2830.
- Z. Quan, Z. Wang, Y. Le, B. Yao, K. Li, and J. Yin. 2019. [An efficient framework for sentence similarity modeling](#). *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):853–865.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [Semeval-2021 task 1: Lexical complexity prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Yan Wang. 2017. [The Influence of “i+1” on Chinese Foreign Language Teaching Methods](#). *Sino-US English Teaching*, 14(8).
- Xunjie Zhu, Tingfeng Li, and Gerard de Melo. 2018. [Exploring semantic properties of sentence embeddings](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 632–637, Melbourne, Australia. Association for Computational Linguistics.

# Stanford MLab at SemEval-2021 Task 1: Tree-Based Modelling of Lexical Complexity Using Word Embeddings

Erik Rozi\*, Niveditha Iyer\*, Gordon Chi, Enok Choe, Kathy Lee,  
Kevin Liu, Patrick Liu, Zander Lack, Jillian Tang†, and Ethan A. Chi†

Stanford University

{erikrozi, nivsiyer}@stanford.edu  
{jiltang, ethanchi}@cs.stanford.edu

## Abstract

This paper presents our system for the single- and multi-word lexical complexity prediction tasks of SemEval Task 1: Lexical Complexity Prediction. Text comprehension depends on the reader’s ability to understand the words present in it; evaluating the lexical complexity of such texts can enable readers to find an appropriate text and systems to tailor a text to an audience’s needs. We present our model pipeline, which applies a combination of embedding-based and manual features to predict lexical complexity on the CompLex English dataset using various tree-based and linear models. Our method is ranked 27 / 54 on single-word prediction and 14 / 37 on multi-word prediction.

## 1 Introduction

The rapid expansion of social media and other online channels has made readable information available at an astounding rate. However, the accessibility of this information is often limited by the complexity of this information, especially among readers with low literacy levels in the language of the text and those with reading disabilities. Furthermore, even to the average reader, specialized jargon found in governmental documents and scientific fields is often difficult to decipher.

Systems to guide these users may redirect readers to more easily comprehensible sources, convert the text to simpler wording, or provide additional information about what difficult words mean. The development of such systems is benefited by the ability to evaluate the complexity of sections of the text. While there is currently a large amount of available text data, very little of it is labeled with word complexity; automating the labelling process would make much more data available to aid the

development of NLP systems in tasks such as text simplification.

Multiple features of a word can affect lexical complexity. In addition to a word’s frequency, length and syllable count, the context in which a word is found is likely to affect its understandability. The additional factor of the reader’s proficiency in a language makes this task complex as many words have a highly variable complexity.

In this paper, we describe our model that predicts single- and multi-word lexical complexity scores.

## 2 Background

### 2.1 Task Overview

All data was provided through SemEval Task 1 (Shardlow et al., 2021). Our dataset consists of an augmented version of the CompLex Corpus (Shardlow et al., 2020), which contains English sentences from three genres of corpora: the Bible, Europarl, and biomedical writing. From each sentence, both single- and multi-word tokens were selected and annotated by approximately 7 annotators. Each token was annotated on complexity from a scale of 1-5, though for this competition, complexity was normalized to a continuous scale between 0 and 1.

Token complexity can differ based on the complexity of the token both with and without context. For example, for one instance, the token *river* was rated to have a complexity of 0.0, while *jurisprudence* had a complexity of around 0.672 for another instance. However, token complexities can also change based on the context from which it came from. For example, the token *wisdom* was given a complexity of 0.125 when it was associated in the sentence “*The rod of correction gives wisdom, but a child left to himself causes shame to his mother.*” However, the same token was given a significantly higher complexity score of 0.368 when associated with the sentence “*For in much wisdom is much*

\*Co-first authors.

†Co-senior authors.

*grief; and he who increases knowledge increases sorrow.”*

Given that GloVe embeddings (Pennington et al., 2014) store semantic meaning of single words, we chose to use GloVe embeddings to represent both tokens and sentences. With this approach, we determine that despite contextual variation, inherent properties of the token itself are sufficient to explain much of the variance in lexical complexity.

## 2.2 Traditional Text Complexity Metrics

Many traditional metrics for calculating the complexity of text predict with syllable to word count ratios. For example, the Flesch-Kincaid Grade Level Formula <sup>1</sup> (Kincaid et al., 1975) calculates the complexity of a text with the formula

$$GL = 0.39 \frac{\text{words}}{\text{sentence}} + 11.8 \frac{\text{syllables}}{\text{word}} - 15.59.$$

Other models based on the grade level of a text, such as the Automated Readability Index and the SMOG Index (Kincaid et al., 1975), also exist. Our original hypothesis inferred that these indexes would be good indicators to predict the complexity of a token. However, through empirical analysis, we found that these indicators provided no marginal benefit compared to GloVe sentence embeddings and simpler handcrafted features. As seen in Table 1, we found that the correlation coefficients of traditional complexity metrics to dataset complexity values were low. To test this, we initially included these traditional metrics in our feature space for the following models. Our model reported an R score of 0.63 with the Flesch-Kincaid Grade and SMOG Index as additional features. We removed these features after observing little benefit or worse loss scores (in comparison to Table 2). This suggests that word complexity in context may be embedded in a deeper semantic level than simple word and syllable lengths.

Model	Pearson
Flesch-Kincaid Grade	0.07
Automated Readability Index	0.07
SMOG Index	0.03

Table 1: Pearson correlation between complexity metrics and true complexity values (single-word)

<sup>1</sup><https://github.com/shivam5992/textstat>

## 3 System Overview

### 3.1 Single Word Complexity Score

#### 3.1.1 Data Representation and Features

This system uses a combination of GloVe (Pennington et al., 2014) word embeddings and hand-crafted features as final features to predict complexity on. Pre-trained GloVe embeddings with a dimension of 300 for both the single-word token and each word in the context sentence were used. For the single-word embeddings, PCA with a final dimension of 100 was applied. Since the context sentences contained a variable number of words, we calculated the component-wise mean of all the word-vector representations in the context sentence. We found that sentence features had low mutual information, hence we decided to use a limited number of 10 PCA features to calculate the mean of the sentence features. This mean representation is concatenated with the GloVe embedding of the single-word token.

In other words, let  $\mathbf{t}$  be the GloVe embedding of the single-word token, and  $\mathbf{w}_i$  be the GloVe embedding for word  $i$  in the context sentence, with  $n$  words. We calculate the sentence representation  $\mathbf{s}$  to be

$$\mathbf{s} = \frac{\sum_i^n \mathbf{w}_i}{n},$$

leading to features  $\mathbf{r} = [\mathbf{t}, \mathbf{s}]$  with a dimensionality of 110 features.

On top of this representation, we include hand-crafted features. Through manual tuning, we created a set of manual features:

- **NUMLETTERS**: the number of letters in the token
- **NUMCAPITALS**: the number of capital letters in the token
- **NUMSYLLABLES**: the number of syllables in the token
- **NUMDIGITS**: the number of digits in the token
- **ISFIRSTCAPITAL**: whether or not the first letter is capitalized (implying it is a subject or technical term)
- **NUMSENTWORDS**: the number of words in the context sentence
- **CORPUSTYPE**: the type of corpus the sentence is taken from

- **POS**: the part of speech of the token
- **ISINNER**: whether or not the token is in a named entity

The "POS" and "IsInNER" features are obtained from the Stanza NLP package (Qi et al., 2020).

Instead of relying on the frequencies of words in the text we were analyzing, we found that a more representative frequency metric could be obtained by counting word occurrences in all Wikipedia articles. Hence we decided to use frequencies of word as they appear in the English Wikipedia articles as of February 2019.<sup>2</sup> This feature was concatenated with all of the other handcrafted features and GloVe embeddings, leading to a final feature dimensionality of 126.

### 3.1.2 Learning Models<sup>3</sup>

Because the system primarily treats the input datapoints as sets of vectors and other numerical features, most of the models used were regressors made for data. As the baseline, we used linear regression with the GloVe embeddings for only the single-word token and obtained a baseline R of 0.7888 on the train set.

We explored the following machine learning models:

- **Ridge regression** is a linear least squares model with L2 regularization to reduce overfitting and variance. We use  $\alpha = 0.00001$  as the regularization coefficient to prevent overfitting.
- **Support Vector Regression** is a Support Vector Machine for a regression task that tolerates errors within a certain degree  $\epsilon$ . We use  $\epsilon = 0.02$  as the distance within which no penalty is associated, and  $C = 0.2$  as a regularization parameter to reduce overfitting.
- **Decision Tree Regression** creates a model as a series of decision rules. As a baseline, we created a decision tree with `max_depth = 6`, though other models use varying depths.
- **AdaBoost Regression** (Freund and Schapire, 1996) sequentially applies decision trees, with each tree placing more weight on data that

<sup>2</sup><https://github.com/IlyaSemenov/wikipedia-word-frequency>

<sup>3</sup>All models were implemented using SKLearn (Pedregosa et al., 2012) unless otherwise mentioned.

previous trees did not fit well to. We use DecisionTreeRegressors with `max_depth= 10` as the base estimator, with a total of  $n_{estimators} = 20$  decision trees.

- **XGBoost Regressor** overcomes the inefficiency in gradient boosting of creating a single decision tree at a time by parallelizing tree building. We used `max_depth= 4` and  $\lambda = 2000$  as a regularization parameter. As  $\lambda$  is responsible for L2 regularization of weights, using a higher value would make the model more conservative by encouraging smaller weights.
- **LightGBM Regressor**<sup>4</sup> (Ke et al., 2017) is a framework that uses tree based learning algorithms for gradient boosting. Our model uses gain-based feature importance, with  $\lambda = 50$  and  $n_{leaves} = 40$  and a minimum of 100 datapoints per leaf. To avoid overfitting, we regularize with path smoothing of 1, set a maximum tree depth of 15, and trained using DART boosting.
- **Stacking** We also tested a stack of estimators with a final Ridge regressor to get an ensemble of predictions and reduce overfitting. We stacked five AdaBoost Regressors with  $n_{estimators} = 50, 100$  estimators respectively, each with a base estimator of a Decision Tree Regressor with `max_depth` varying between 5, 7, and 9. On top of this, we stacked two Support Vector Regressors with  $\epsilon = 0.01, 0.001$  and  $C = 0.1, 0.01$  respectively. Finally, we stacked three LightGBM Regressors, each with 100, 50, and 10 leaves respectively. This method was used with the theory that combining multiple models would result in better predictive power than one model alone.
- **Bagging** is an ensemble method involving training copies of a base model on independent random samples of the full dataset. We used an LGBM with  $n_{leaves} = 40$ , `reg_lambda = 100`, `path_smooth = 1`, `max_depth = 12`, and `feature_fraction = 0.75` as our base model. We set  $n_{estimators} = 10$ , `max_samples = 0.8`, and `max_features = 0.75` in order to reduce variance of the decision tree.

<sup>4</sup><https://github.com/microsoft/LightGBM>

- **BERT** We also explore context-dependent deep learning architectures: in particular, we fine-tune the pre-trained BERT model (Devlin et al., 2019). We leverage the pre-trained BERT neural network<sup>5</sup> by tokenizing each sentence, and providing the target word to the model as a second sentence. With 2-3 fully connected layers added on top of the pre-trained model, we fine-tuned this model to generate a numerical complexity prediction, by optimizing on the L2 Loss. All experiments were implemented using SKLearn (Pedregosa et al., 2012) and HuggingFace<sup>6</sup>.

## 3.2 Multi-word Complexity Score

### 3.2.1 Data Representation and Features

Our multi-word data representation closely mirrored our single-word token representation. All hand-crafted features were crafted in the same way as the single word counterparts, except for the POS and NER features which were not included. For example, the feature **NumLetters** includes the number of letters from both words. The context sentence embeddings were calculated with the same methodology of applying PCA with dimension of 10 to the mean of the GloVe embeddings.

The key difference between the two models lies in the representation of the multi-word tokens themselves. The data provided was consistent in that each multi-word token consisted of two words. Therefore, to represent these tokens, we concatenated the GloVe representation of each word in the token, as well as the difference between both GloVe vectors. From there, we applied PCA of dimension 150 to this embedding, which was determined through experimentation, and concatenated this with the other hand-crafted and context sentence features mentioned previously.

More concretely, let  $\mathbf{t}_1, \mathbf{t}_2$  be the GloVe embeddings of each word in the multi-word token. We found the new representation of a multi-word token  $\mathbf{m}$  to be

$$\mathbf{m} = [\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_1 - \mathbf{t}_2].$$

This was concatenated with sentence representation  $\mathbf{s}$  and handcrafted features for a final dimensionality of 174 features.

<sup>5</sup>We use tokenizers and pre-trained models from the HuggingFace transformers library: [https://huggingface.co/transformers/model\\_doc/bert.html](https://huggingface.co/transformers/model_doc/bert.html)

<sup>6</sup><https://huggingface.com>

### 3.2.2 Learning Models

Given the similarity of the multi-word representations versus the single-word representations (the only difference being the addition of a second token's GloVe embedding), we used the LightGBM Regressor outlined in section 3.1.2, as this model performed the best in the single word token setting. This proved to be an effective way to predict multi-word complexity.

## 4 Experimental Setup

The train and validation dataset splits provided were used in our experimental setup. In addition, we used K-fold validation to reduce overfitting. Using K-fold, we split the training set into  $k$  smaller sets arbitrarily, train using  $k - 1$  folds, and cross-validate with the remaining fold in the train set. This reduces leakage from the validation set into the model so that we can accurately validate our methods.

Task predictions were evaluated using Pearson correlation, though Spearman correlation, mean absolute error, mean squared error, and R-squared were also reported. We compared the performance of our own models using Pearson correlation to keep one consistent evaluation metric.

## 5 Results

### 5.1 Single Word Results

From Table 2, LGBMRegressor performs the best in terms of the Pearson metric. Therefore, we chose this model as our final model for submission.

We found that transforming the word frequencies to a logarithmic scale did not improve results across the models we tested. This is expected because tree-based regressors (Adaboost, LGBM, XGB) are invariant to monotonic scaling. Our results on the task evaluation metrics are shown in Table ??.

We suspect Ensemble Stacking overgeneralized and did not perform effectively as a result, though other stacking methods could perform better. Surprisingly, the contextual deep learning approach of BERT did not perform well on the task, only approaching similar performance to the baseline linear regression on GloVe embeddings.

Though we scored 27th place out of 54 teams overall in the Pearson metric for single-words, the top score was only 0.03 points higher than our own evaluation score. We suspect that different methods of stacking regressors and using complex decision

Model	Pearson
Linear Regression	0.7888
BERT	0.7892
Ridge	0.7829
SVR	0.7945
DecisionTreeRegressor	0.7083
AdaBoost Regressor	0.7976
Ensemble Stacking	0.7578
XGBRegressor	0.7884
BaggingRegressor	0.8018
LGBMRegressor	<b>0.8056</b>

Table 2: Experimental results (single-word)

Metric	Score	Ranking
Pearson	0.7533	(27)
Spearman	0.7044	(34)
MAE	0.0653	(25)
MSE	0.0071	(29)
R2	0.5615	(26)

Table 3: Evaluation results (single-word)

trees would have created a model that predicts well with the CompLex dataset. However, whether this type of model will generalize to future datasets is a subject of investigation.

## 5.2 Multi-word Expressions Results

We note that our multi-word expression Pearson metric, as shown in Table ??, performs better than our single word Pearson, and ranks 14th out of 37 teams. This is most likely because averaging the GloVe representations of the two tokens allows for more data points to be represented in the decision tree model.

## 6 Conclusion

In this paper we describe tree-based modelling of words in context to predict lexical complexity. We find that lexical complexity is already embedded in

Metric	Score	Ranking
Pearson	0.8280	(14)
Spearman	0.8124	(18)
MAE	0.0711	(24)
MSE	0.0080	(24)
R2	0.6671	(14)

Table 4: Evaluation results (multi-word)

GloVe representations of words and that complex architectures provide some increase in predictive performance.

For future work, we suggest taking additional contextual features into account, such as the proximity of each neighboring word. We also suggest looking into newer transformer models to represent contextual embeddings.

As larger bodies of text become widely available to wide audiences for public consumption, we are hopeful that such systems will help readers identify suitable texts for their reading level and help build systems that can tailor text to varied reading levels, allowing for greater accessibility.

## Acknowledgments

This research effort would not have been possible without the support of Stanford ACMLab. The authors thank Matthew Shardlow, Richard Evans, Gustavo Henrique Paetzold and Marcos Zampieri for organizing SemEval 2021 Task 1: Lexical Complexity Prediction. We also thank Yasmine Mitchell for helpful discussions.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning, ICML'96*, page 148–156, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. [Lightgbm: A highly efficient gradient boosting decision tree](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- J. Peter Kincaid, Robert P. Jr. Fishburne, Richard L. Rogers, and Brad S. Chissom. 1975. [Derivation of new readability formulas \(automated readability index, fog count and flesch reading ease formula\) for navy enlisted personnel](#). *Institute for Simulation and Training*.



Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2012. [Scikit-learn: Machine learning in python](#). *CoRR*, abs/1201.0490.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. [Stanza: A Python natural language processing toolkit for many human languages](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [CompLex — a new corpus for lexical complexity prediction from Likert Scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. Semeval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

# archer at SemEval-2021 Task 1: Contextualising Lexical Complexity

Irene Russo

ILC-CNR, Pisa, Italy

irene.russo@ilc.cnr.it

## Abstract

Evaluating the complexity of a target word in a sentential context is the aim of the Lexical Complexity Prediction task at SemEval-2021. This paper presents the system created to assess single words lexical complexity, combining linguistic and psycholinguistic variables in a set of experiments involving random forest and XGboost regressors.

Beyond encoding out-of-context information about the lemma, we implemented features based on pre-trained language models to model the target word's in-context complexity.

## 1 Introduction

Lexical complexity prediction is the task aiming at evaluating the complexity of a word in context, modeling a crucial aspect of reading comprehension. Complex words can slow down the reading process; there is a well-known correlation between a word's difficulty and the time spent looking at it, as emerging from eye-tracking experiments (Mousikou et al., 2021).

Assessing the complexity of a specific word in context is a crucial prerequisite for NLP systems aiming to evaluate a text's readability and produce a simplified version of it. It is a prerequisite for text simplification systems based on lexical substitutions and can be the starting point to tailor a text to the user's needs. It is a topic worthy of investigation from multiple points of view. In the past, datasets containing crowdsourced evaluations of lexical items' complexity (Paetzold and Specia, 2016b; Štajner et al., 2018) have been used in evaluation campaigns.

In this paper, we introduce the system used to assess single English words lexical complexity at SemEval-2021 Lexical Complexity Prediction task (Shardlow et al., 2021). Based on previous approaches to this issue, we combine linguistic and

psycholinguistic variables, using a random forest regressor and an XGboost regressor. The majority of the variables encode information about the word out-of-context (e.g., frequency, number of letters, age of acquisition) without considering the sentential context and how it can affect an item's complexity.

To approximate in-context complexity, we consider the cloze probability of a word as its probability to complete a particular sentence frame. We experiment with different language models in a masked word prediction framework, taking into account the first ten most probable words occurring in that context.

## 2 Related works

A wide range of approaches has been used for lexical complexity prediction in past evaluation campaigns. However, previous tasks focused on classification since the proposed datasets labeled words in context as easy or difficult.

Including more classes makes the task more difficult. (Garí Soler et al., 2018) investigate the role of word embeddings in lexical complexity prediction for French words, using as training sets two French lexical resources that encode the distribution of words across different levels of difficulty. According to the authors, the task is influenced by the context of use of the words. Word embeddings, encoding contextual information, can be helpful to determine the lexical complexity of target words. The authors experimented with different neural network settings (with and without hidden layers), using as features the number of characters, the number of phonemes, and the log frequency in a corpus of film subtitles plus word embeddings trained on Wikipedia with fastText. However, the combination of word embeddings with such features does not improve the results compared with other sets

of features that always contain frequency, a crucial indicator of lexical complexity.

In past evaluation campaigns, there have been occasional attempts at incorporating word embeddings models in the automatic evaluation process, with the assumption that lexical complexity should be evaluated as a contextual variable. However, better results have been obtained considering just static, out-of-context properties of lemmas.

More recently, the trend to use embeddings from language models to predict psycholinguistic variables can provide insights about how to incorporate them in experiments aiming at understanding the complexity of human comprehension (Hao et al., 2020).

However, if we frame lexical complexity as a measure strongly dependent on words' psycholinguistic properties, we should recognize that past computational efforts for predicting word norms did not take into account the role of context (Russo, 2020; Charbonnier and Wartena, 2019). Static word embeddings such as word2vec have been used to predict values of psycholinguistic norms usually assessed in experimental settings (Ljubešić et al., 2018; Rothe and Schütze, 2016). More recent Transformed-based language models that consistently incorporate contextual knowledge have not yet been considered for this task.

### 3 The Dataset

The Lexical Complexity Prediction shared task at SemEval-2021 (LCP-2021) (Shardlow et al., 2021) is based on an English dataset with a 5-point Likert scale annotation. The complexity score is similar to that included in another dataset (Shardlow et al., 2020). It ranges from very easy for very familiar words to very difficult (unclear words that an annotator had never seen before). Annotators are explicitly invited to evaluate the role of the sentence in inferring the meaning of the word. The task is structured into two sub-tasks:

- Sub-task 1: predicting the complexity score of single words;
- Sub-task 2: predicting the complexity score of multi-word expressions.

The task is inspired by two previous competitions (CWI 2016 and CWI 2018) about boolean complex word identification, aiming at identifying which words are likely to be considered complex or

domain	mean	std dev
bible	0.296	0.132
europarl	0.287	0.109
biomed	0.325	0.152

Table 1: Mean complexity and standard deviation for each domain in the LCP-2021 training dataset.

not by a given target population. However, in LCP-2021 lexical complexity is a continuous property, and the task consists of predicting the complexity score for each target word in context.

7,662 sentences and 3,298 unique tokens compose the LCP-2021 training dataset for single words evaluation: each token appears in more than one sentence, making the impact of context crucial especially for subsets of sentences with highly variable values. For example, the word *livers* occurs 2 times in the dataset, with different complexity scores:

- *The activity of BCKDH in livers of homozygous knockout mouse pups was undetectable, accounting for the accumulation of unmetabolized BCAA.* (complexity score = 0.0499)
- *The comparison of gene expression in livers of mock- or cadmium-treated Mtf1Mx-cre and Mtf1loxP mice revealed several MTF-1 target gene candidates.* (complexity score = 0.323)

Sentences are extracted from three domains: the Bible, the English part of the European Parliament proceedings, and a biomedical corpus composed of scientific papers. Table 1 reports the mean complexity and the standard deviation for each domain. Target words extracted from the biomedical corpus are the most complex. Due to the variability in complexity for the same target word, the biomedical corpus is also the domain that poses significant challenges.

We propose a system for sub-task 1, encoding for each target word numerical values concerning a set of variables described in Section 4. We do not propose a system for sub-task 2.

### 4 Out-of-Context and In-Context Lexical Complexity

The lexical complexity of a word can be represented as an out-of-context property of a lemma or an in-context property of a word.

Following the first approach, the same lemma

has a fixed lexical complexity value, depending on features such as the number of characters or senses in WordNet (see the list of out-of-context features below).

When considering the word in context, its disambiguation could affect its complexity rating because a sense could be more complex than the others (for example, when a word is used in its specialized sense). However, because of the lack of methodologies for assessing senses' complexity and the unsatisfactory performance of word sense disambiguation systems, the role of senses' complexity for lexical complexity prediction can not be investigated. Several systems participating at CWI2018 took into account the role of context, focusing on the whole sentence where target items occur. However, quite interestingly, one of the best models (Gooding and Kochmar, 2018) does not consider the influence of the textual context for determining the target word's complexity. We provide the list of out-of-context features used in our system:

- Length: length of each target word (number of characters);
- Syllables: number of syllables of each target word<sup>1</sup>;
- length sentence: length of each sentence (number of tokens);
- Word freq: frequency of the target word in the Exquisite Corpus<sup>2</sup>;
- AoA Kup: age of acquisition (AoA) of the target word in (Kuperman et al., 2012) dataset. The age of acquisition of a word is a psycholinguistic variable concerning the age at which a word is typically learned. We assume that easy words are learned at a younger age;
- Children freq: the natural logarithm of the frequency of lemmas in children movies subtitles included in a corpus of subtitles (Paetzold and Specia, 2016a). We expect that difficult words will be less frequent in this corpus;
- Visual Genome (VG) freq: the natural logarithm of the lemmas' frequency in the Visual Genome descriptions corpus. The Visual Genome dataset (Krishna et al., 2017)

<sup>1</sup>The values are obtained using syllables 0.1.0 <https://pypi.org/project/syllables/>

<sup>2</sup>The values are obtained using wordfreq 2.3.2 <https://pypi.org/project/wordfreq/>

is the largest dataset of image descriptions for English. It is composed of dense annotations of objects, attributes, and relationships between objects for 108K images. As a pre-processing step, the descriptions have been annotated with TreeTagger (Schmid, 1994) and the list of lemmas has been ordered by frequency;

- ImageNet: the presence of the target word in ImageNet (boolean feature). Only concrete nouns can be included as pictures in this resource. We assume that easy words tend to be more frequently concrete (Russakovsky et al., 2015);
- Uppercase: this variable takes into account the relative number of uppercase letters in the target words, and it is used to detect acronyms. We notice that acronyms are generally rated as difficult word;
- Scrabble: for each target word, its value according to Scrabble's rules. In this word game, each letter's number of points is based on the letter's frequency in standard English. We expect that complex words will have higher ratings;
- Senses: number of senses of the target word in WordNet (Fellbaum, 1998);
- Bible/europarl/biomed: boolean feature encoding if the sentence belongs to one of these domains.

A word  $x$  is difficult in a sentential context if the reader has never encountered it. The relative frequency should be sufficient to explain the perceived complexity of  $x$ ). A word can also be difficult if its meaning in that specific context is not the most common one, i.e when a specialized sense is accessed or a metaphorical meaning is created. If we know a word by the company it keeps, we do not know a word when its company is somewhat eccentric.

Lexical complexity as an in-context property can be modeled considering the influence of the surrounding text. There are two ways to model the textual context's influence on the lexical complexity of a target word: local context (a window span surrounding the target word) and global context (the whole sentence). In the first case, words surrounding the target word can increase the overall

complexity in that text span. In the second case, the probability of a word in a masked word prediction task that concerns the whole sentence can be a good approximation of the intuition that words semantically difficult to generate are more complex than the simpler ones. We implemented two types of variables as in-context features to address in-context lexical complexity:

- Position [Language Model]: The target word’s position among the first ten most probable words completing the sentence in a masked context for five language models. The language models tested are BERT, XLNet large, BART, ELECTRA, and RoBERTa. We used pre-trained models made available by HuggingFace;
- Out-of-context complexity of the previous tokens: the value is obtained by selecting the five content words (nouns, adjectives, or verbs) preceding the target word and averaging their complexity values resulting from a random forest regressor that includes just out-of-context variables.

The Pearson correlations among each feature and target word’s lexical complexity reveal that word frequencies are the most relevant features, especially frequencies extracted from children movies’ subtitles (see Figure 1). The age of acquisition of words is another variable strongly correlated with the complexity of the target words ( $r=0.55$ ).

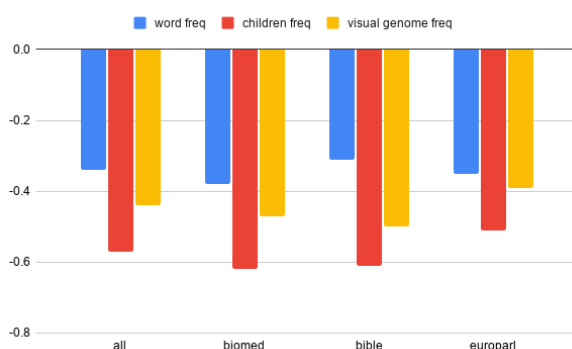


Figure 1: Pearson correlations between lexical complexity and word frequencies from different corpora, reported for each LCP-2021 domain.

## 5 Experiments

The set of features described in Section 4 has been implemented for the training set, tested on trial set,

Domain	MAE	R
bible	0.075	0.69
europarl	0.054	0.76
biomed	0.066	0.86
all_out_of_context	0.063	0.817
all_in_context	0.095	0.415
all	0.065	0.80

Table 2: Random forest regression results on trial set.

features	MAE	R
all_out_of_context	0.063	0.793
all	0.062	0.799

Table 3: Average random forest regression results on five training-trial splits.

and - to avoid overfitting - on multiple training-test splits with test sets mimicking the trial set’s composition. We experimented with a system based on a random forest regressor (RF), and a system based on an XGboost regressor (Chen and Guestrin, 2016), both implemented in sklearn. For the RF regressor, we choose to measure the quality of a split with mean absolute error. We also normalized the features with the standard scaler function (scaling each feature between 0 and 1). We obtained comparable results, with random forest regressor performing slightly better for several training-trial splits. For this reason, Table 2 summarises RF results. We report the mean absolute error (MAE) and Pearson correlation (R), used to rank the systems at LCP-2021 task.

In-context features emerge as useless from these results; however, testing with different trial sets, we infer that this set of features could improve the performance (see Table 3) and, as a consequence, we included all the features for the processing of the test set released by LCP-2021 organisers. Concerning the role of word frequencies, that are negatively correlated with lexical complexity (see Section 4), frequencies from a general corpus used together with frequencies from children movies subtitles guarantee a good performance of the RF regressor in terms of MAE and Pearson correlation (see Table 4). Our system ranked 22 out of 54 for the single word complexity prediction task. The best result on the test set was obtained using all the features and the random forest regressor (see Table 5).

features	MAE	R
Word freq + children freq	0.069	0.74
Word freq + VG freq	0.068	0.752

Table 4: Average random forest regression results on five training-trial splits for frequency features.

all features	
Pearson	0.7561
MAE	0.0641
Spearman	0.7067
MSE	0.0069
R2	0.5707

Table 5: Best random forest regression results on test set (official results).

## 6 Conclusions

This paper briefly reports the system created to predict single words’ complexity score for the Lexical Complexity Prediction shared task at SemEval-2021 (LCP-2021).

Our system ranked 22 out of 54 for this sub-task, with slightly inferior results to the ones obtained on trial sets. The significative role of frequencies extracted from different corpora paves the way to further investigations in this direction.

Encoding in-context complexity as a variable related to pre-trained language models’ predictions had no significant impact on the results. However, in-context complexity could be modeled in different ways. Experimenting with how in-context target word’s complexity changes depending on the frequencies of the surrounding words is a future analysis topic.

## References

Jean Charbonnier and Christian Wartena. 2019. [Predicting word concreteness and imagery](#). In *Proceedings of the 13th International Conference on Computational Semantics - Long Papers*, pages 176–187, Gothenburg, Sweden. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 2016. [Xgboost: A scalable tree boosting system](#). KDD ’16, page 785–794, New York, NY, USA. Association for Computing Machinery.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Bradford Books.

Aina Garí Soler, Marianna Apidianaki, and Alexandre Allauzen. 2018. [A comparative study of word](#)

[embeddings and other features for lexical complexity detection in French](#). In *Actes de la Conférence TALN. Volume 1 - Articles longs, articles courts de TALN*, pages 499–508, Rennes, France. ATALA.

Sian Gooding and Ekaterina Kochmar. 2018. [CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.

Yiding Hao, Simon Mendelsohn, Rachel Sterneck, Randi Martinez, and Robert Frank. 2020. [Probabilistic predictions of people perusing: Evaluating metrics of language model performance for psycholinguistic modeling](#). In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 75–86, Online. Association for Computational Linguistics.

Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2017. [Visual genome: Connecting language and vision using crowdsourced dense image annotations](#). *Int. J. Comput. Vision*, 123(1):32–73.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. [Age-of-acquisition ratings for 30,000 english words](#). *Behavior Research Methods*, (44):978–990.

Nikola Ljubešić, Darja Fišer, and Anita Peti-Stantić. 2018. [Predicting concreteness and imageability of words within and across languages via word embeddings](#). In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 217–222, Melbourne, Australia. Association for Computational Linguistics.

Petroula Mousikou, Lorena Nüesch, Jana Hasenäcker, and Sascha Schroeder. 2021. [Reading morphologically complex words in german: the case of particle and prefixed verbs](#). *Language, Cognition and Neuroscience*, 36(2):255–268.

Gustavo Paetzold and Lucia Specia. 2016a. [Collecting and exploring everyday language for predicting psycholinguistic properties of words](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1669–1679, Osaka, Japan. The COLING 2016 Organizing Committee.

Gustavo Paetzold and Lucia Specia. 2016b. [Semeval 2016 task 11: Complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*.

Sascha Rothe and Hinrich Schütze. 2016. [Word embedding calculus in meaningful ultradense subspaces](#). In *Proceedings of the 54th Annual Meeting of the*

*Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–517, Berlin, Germany. Association for Computational Linguistics.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. [ImageNet Large Scale Visual Recognition Challenge](#). *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Irene Russo. 2020. [Guessing the age of acquisition of Italian lemmas through linear regression](#). In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 43–48, Online. Association for Computational Linguistics.

Helmut Schmid. 1994. [Probabilistic part-of-speech tagging using decision trees](#). In *Proceedings of International Conference on New Methods in Language Processing*.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [Complex: A new corpus for lexical complexity prediction from likert scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [Semeval-2021 task 1: Lexical complexity prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.

# katildakat at SemEval-2021 Task 1: Lexical Complexity Prediction of Single Words and Multi-Word Expressions in English

**Katja Voskoboïnik**

Aalto University

Helsinki, Finland

ekaterina.voskoboïnik@aalto.fi

## Abstract

This paper describes systems submitted to SemEval 2021 Task 1: Lexical Complexity Prediction (LCP). We compare a linear and a non-linear regression models trained to work for both tracks of the task. We show that both systems are able to generalize better when supplied with information about complexities of single word and multi-word expression (MWE) targets simultaneously. This approach proved to be the most beneficial for multi-word expression targets. We also demonstrate that some hand-crafted features differ in their importance for the target types.

## 1 Introduction

SemEval-2021 Task 1 is the task of Lexical Complexity Prediction (LCP) (Shardlow et al., 2021). The goal of the task is to assign a target in a context a continuous value ranging between 0 and 1, where 1 indicates complete unintelligibility and 0 signals perfect familiarity as perceived by a native speaker. The task has two tracks: predicting the complexity score of single words and predicting the complexity score of multi-word expressions (MWE). Such a task can be useful in applications like text simplification or automatic language proficiency evaluation.

The CompLex dataset (Shardlow et al., 2020) used in this task is the first English dataset for the task of LCP. The dataset contains single words and MWEs annotated with their lexical complexity score in a specific context. The annotations were provided by native speakers of English. The targets and their contexts were obtained from texts of different domains: the Bible, Europarl, and biomedical texts. The dataset opens several avenues for research. For instance, how does the perceived complexity of single words and MWEs differ? How does context affect the lexical difficulty of a target?

How does text genre affect comprehensibility of words?

We were interested if there is a difference in performance when using the same representation methods for single words and MWEs. It was decided to approach both tracks as the same problem. We did not distinguish between MWEs and single words and they both were treated as one lexical unit. The same array of features was extracted to represent the targets and linear and non-linear regressors were trained using both subcorpora. This strategy showed performance gains for both single targets and MWEs.<sup>1</sup>

In addition, we wanted to investigate how much the classic hand-crafted features like frequency and length together with subword information and contextualized embeddings (not employed previously for LCP) contribute to complexity estimation of both single words and MWEs. We present the analysis of feature importance rankings in Section 6.

## 2 Related Work

Complex Word Identification (CWI) is the task of determining how difficult a lexical unit is to a target audience (Shardlow, 2013). The knowledge about the lexical unit complexity can benefit several NLP tasks such as text simplification (TS) or applications related to second-language (L2) acquisition.

The goal of TS is to adapt a text to make information, for example, news, more accessible for readers. The TS target group can be language learners (Petersen and Ostendorf, 2007), people with cognitive disabilities (Yaneva et al., 2016) or people with low literacy skills (Aluisio et al., 2010). One of the strategies of TS is lexical simplification (LS). LS is the task of substituting complex words with simpler ones without changing the original mean-

<sup>1</sup>The code and notes are available at <https://github.com/katildakat/COMPLEX>



ing. To perform the LS one should first identify the units that might pose a difficulty (Shardlow, 2013).

In the area of L2 acquisition, lexical complexity information can be used both for generating study materials appropriate for a learner’s level (Alfter and Volodina, 2018) and for evaluating how proficient a student is (del Río, 2019).

To understand what makes a word complex for a target audience, one needs to obtain data with complexity annotated. The first manually labelled resource for CWI (CWI 2016) was introduced in SemEval-2016 Task 11 (Paetzold and Specia, 2016). It contained sentences with words marked by non-native speakers of English as either difficult or easy to understand. A word was labelled as complex if at least one annotator marked it as such. Thus, words were classified in a binary fashion without addressing the proficiency levels or native languages of the annotators.

Another CWI dataset (CWI 2018) was presented for BEA workshop 2018 (Yimam et al., 2018). It contains CWIG3G2 datasets (Yimam et al., 2017) expanded further with the French subcorpus. This is a multilingual dataset with words and MWEs marked as complex or simple within a given context. There is no standard form for MWEs. The annotators were free to label any sequence of words as a difficult MWE. The complexity judgments were collected from native and non-native speakers. In addition to the binary labels, the words were also assigned an aggregated complexity score. The score was computed as the proportion of annotators that found a word complex.

In summary, in both CWI 2016 and CWI 2018 the annotators were not asked to provide a degree of difficulty. The MWEs in CWI 2018 were not clearly defined making the nature of their complexity hard to investigate. The CompLex dataset used in SemEval-2021 Task 1 was constructed to amend the aforementioned faults of CWI 2016 and CWI 2018. First, it treats complexity as a continuous value. Second, it bounds MWEs to only pairs of adjective-noun or noun-noun phrases allowing for more targeted research. We believe that models trained using CompLex have more flexibility in their application. For example, one could set a threshold of complexity to account for different language proficiency levels for both TS and L2 acquisition-related applications.

One of the goals of CWI is to establish what makes a word complex. The reports for CWI 2016

(Paetzold and Specia, 2016) and CWI 2018 (Yimam et al., 2018) as well as the investigation of CWI 2016 results (Paetzold and Specia, 2016) show that such features as frequency and length are the most predictive for establishing word complexity. Moreover, according to the baselines provided by SemEval-2016 Task 11 organizers (Paetzold and Specia, 2016), the degree of polysemy of a word was also quite successful. In addition to hand-crafted features, the teams in both competitions made use of different static word embeddings but they didn’t outperform the frequency-based features.

### 3 System overview

A linear and a non-linear models were compared. We have trained a linear regression and a multilayer perceptron using the same array of features. The features can be divided into two categories: embeddings and hand-crafted features for both target and context.

#### 3.1 BERT Embeddings

For embeddings, it was decided to represent targets and their context using BERT model (Devlin et al., 2018). First, BERT is able to provide a target with a representation dependent on the context. Second, because of its next sentence prediction objective during training, it is also able to produce a separate representation for the whole sentence in the same vector space as a target. We were interested to see if target representation would benefit from combination with this additional context information.

When a single word target is present in the BERT’s token vocabulary, it is represented simply as a vector assigned to it by the model. In the case when a target is absent from the BERT’s vocabulary, it is represented as an average vector of its subword embeddings. MWE targets were always represented as an average representation for their BERT tokens. Contexts were assigned with [CLS] token embeddings. Finally, target and context embeddings were combined into a mean vector of 768 dimensions. When used as features to represent the training dataset in the linear regression model, mean embeddings demonstrated a slightly better performance on the trial data than concatenated vectors. For this reason, we have opted for the average embeddings of targets and contexts instead of the concatenation.

### 3.2 Hand-crafted features

In addition to contextualized embeddings, we were also interested to explore other features. We were especially interested to study how the target’s subword information can be used to explain its complexity value. The final set of features was as follows:

1. The number of BERT vocabulary tokens (an average number for MWE) in a target. This feature was chosen because it implicitly contains frequency information about a target. BERT uses WordPiece tokenizer (Wu et al., 2016). WordPiece is a frequency-based word segmentation algorithm. It learns to unite substrings into new vocabulary items to increase the likelihood of its training data. This means the targets that were tokenized into several BERT tokens were infrequent in the tokenizer’s training data.
2. A BERT score for a masked target. This feature was intended to convey information about how easy it is to predict a target in a given context. This approach however has a downside for our specific BERT implementation: BERT base model was trained to predict a randomly masked WordPiece token not a whole word token. It was decided that all single word targets are to be replaced with one MASK token. An average log probability to appear in place of a mask for every target subtoken was collected. MWE targets were substituted with two MASK tokens. An average log probability for subtokens of both words is collected, summed and divided by two.
3. A number of subwords a target is divided into (an average number for MWE) by a Morfessor segmentation model (Virpioja, 2013). This feature was expected to be a better complexity predictor than a target length in characters since it might be able to indicate a number of word parts connected to semantic or grammatical meaning.
4. An average frequency of subwords a target contains. This feature was expected to reflect how easy it would be to derive a meaning from word subparts. The frequencies were estimated using the segmentation model. In CWI 2018 the character n-gram frequency information employed by (Alfter and Pilán, 2018) achieved high results. The success of this approach might be supported by the evidence that morphological awareness affects how both native and non-native speakers process words (Kimppa et al., 2019) (Deacon et al., 2014). This feature was chosen to investigate if frequency of morpheme-like subwords is also a good lexical complexity predictor.
5. A number of WordNet synsets (Fellbaum, 1998) that a target is present in. This feature was used to provide the information on the target’s degree of polysemy. For the MWEs, we counted both synsets for the whole expression as well as synsets where either of the parts is present.
6. The length of a target in characters (an average length for MWEs).
7. Finally, we have chosen to include word frequency (an average frequency for MWEs). Frequency information is known to be a good predictor for complexity, so it was reasonable to use it as a baseline to compare other features to.

For the submitted system, the embedding features were concatenated with hand-crafted features into 775 dimensional vector. This vector was used as an input to both regressor types.

We have also investigated if the described setup would benefit from feature selection. We decided to half the original feature vector’s size in half by leaving only 400 most informative dimensions. For the linear model, the dimensions were ranked by their F-score. The mutual information was used to chose the dimensions for the neural model.

## 4 Experimental setup

### 4.1 Data

During the system development phase, models were trained only with the train subsets of the data, and then their performance was evaluated on trial subsets. For the final submission, both linear and non-linear models were trained with all the data available (single target train and trial, MWE target train and trial).

	LR all	NN all	LR 400	NN 400
Singles	0.666	<b>0.712</b>	0.688	0.707
MWEs	0.783	0.785	<b>0.796</b>	0.774

Table 1: Joint System Results

## 4.2 Parameters and Tools

Both linear regression and neural network models were trained with scikit-learn 0.24.0<sup>2</sup>. It was also used for the feature selection process. The neural network model is a simple Multilayer Perceptron regressor with one ReLu layer of 20 neurons and alpha parameter set to 0.9. Using 8-fold cross validation procedure on all labelled data, we noticed that smaller layer sizes and larger alpha parameters produced better results. However, we feel it is important to note that the hyperparameters were not tuned with proper care, and we believe that better configurations might be possible.

We used BERT base model (cased) to get contextualized embeddings. The cased model was chosen because in CompLex dataset case plays an important role when distinguishing a target from other words in context. The model was used through the 4.0.0 version of transformers library (Wolf et al., 2020).

The Morfessor segmentation model was trained with Morfessor 2.0<sup>3</sup> using logarithmic frequency dampening for words in the data, and with the corpus weight parameter  $\alpha = 0.1$ . The text used for the segmentation model comes from samples of all subcorpora in The Corpus of Contemporary American English (COCA) but for the Academic texts<sup>4</sup>.

The WordNet was used via NLTK 3.4<sup>5</sup>. The word frequencies were obtained using the Zipf frequency estimates in the 'best' wordlist of wordfreq library<sup>6</sup> (Speer et al., 2018).

## 5 Results

The results of the systems trained jointly with single and MWE targets are presented in Table 1. The results for the systems trained with each subcorpus separately are reported in Table 2. The results in both tables are given using Pearson correlation co-

<sup>2</sup>scikit-learn <https://scikit-learn.org/stable/index.html>

<sup>3</sup>Morfessor 2.0 [morfessor.readthedocs.io](http://morfessor.readthedocs.io)

<sup>4</sup>COCA samples [www.corpusdata.org/formats.asp](http://www.corpusdata.org/formats.asp)

<sup>5</sup>NLTK <https://www.nltk.org/>

<sup>6</sup>wordfreq <https://pypi.org/project/wordfreq/>

	LR	NN
Singles	0.669	<b>0.706</b>
MWEs	0.678	<b>0.752</b>

Table 2: Separate Systems Results

FEATURE	S+MWE		S		MWE	
	m	f	m	f	m	f
len_tok	1	1	1	1	1	1
bert_prob	2	2	2	2	3	2
morf_len	3	3	4	3	2	3
morf_freq	9	89	8	53	109	338
n_senses	5	13	3	4	5	5
len_char	4	110	5	197	8	11
word_freq	0	0	0	0	0	0

Table 3: Importance Ranks for Hand-crafted Features

efficients for the test data indicated by row names. LR stands for the linear regression model, and NN stands for the neural regressor. Captions 'all' and '400' distinguish between models trained using all 775 dimensions or 400 with the best scores.

For the single word track, the CodaLab system for some reason accepted only the linear scores, and for the MWE track CodaLab, conversely, accepted only non-linear model predictions. Moreover, the top score for the linear model in the single word track was reported without using Morfessor features.

## 6 Discussion

As can be seen from the results tables, two trends are obvious: MWE targets always benefit from being trained together with single word targets, and the non-linear model tends to slightly outperform the linear one. This can be contributed both to the nature of MWE and to the smaller size of the MWE subcorpus. Although the linear system trained with only single word targets showed better results than the joint one, the non-linear model has also gained from the information about both types of targets when predicting single word complexity. Finally, the feature selection procedure was able to improve the performance of the linear model.

We were interested to find what features for MWEs and single words signal lexical complexity in a similar manner and what features differ in their usefulness. For this purpose, we collected F-scores and mutual information values for the hand-crafted features evaluated for the joint dataset as well as

for the target type subcorpora.

The ranks of hand-crafted features according to their F-scores and mutual information values can be found in Table 3. The features are listed in the same order as they were presented in Section 3.2. The names of the columns reflect the content of the dataset divisions that were explored: S stands for the single targets part of the data, MWE stands for the examples with only MWE targets, and S+MWE marks the results for the joint dataset. The mutual information ranks can be found in 'm' columns and F-scores are given in 'f' columns. The ranks are reported for the features evaluated with the train and the trial parts of the corpus simultaneously.

The information collected in Table 3 shows some differences in feature importance for predicting single word targets and MWE targets, as well as, differences in how suitable some features are for linear and non-linear models. Moreover, the consistency of how high most of the hand-crafted features rank indicates that they remain relevant for LCP and CWI tasks even in presence of such modern approaches as contextualized embeddings.

All hand-crafted features were present in the top 20 highest scored dimensions with mutual information for the joint data and for the single target data. For the MWE targets, information about morph frequency played a less important role placed at only 110 place. Moreover, with the F-score rankings, morph frequency was absent in the top 20 dimensions from all the data configurations.

Target length in characters has not appeared in the top 20 most correlated features for the joint dataset and for the single targets, but it was still relevant for MWEs. Word frequency was rated as the highest correlated feature in all setups, it was followed by the BERT token number feature. These two features were followed by morph number and by the probability of a masked target. Surprisingly, the subword frequency feature was not as successful. The reason for this can be the small amount of data it was estimated on.

## 7 Conclusion

This paper presents the results of two systems submitted to SemEval 2021 Task 1: Lexical Complexity Prediction (LCP). We show that training a system jointly with single word targets and MWE targets benefits the predictability of both target types, especially the MWEs. We also show that the frequency of subwords feature is more predictive for

single targets, while length of a target in characters is more useful for MWE complexity estimation. Finally, we show that classic frequency feature is still the most predictive one, even when used together with new contextualized embeddings.

For the future work, we would like to explore if the underwhelming results of the subword frequency feature can be amended by collecting statistics from a larger resource. Another thing we would like to research is what makes the joint training with single and MWE targets successful. Is it the smaller amount of data available for MWEs? Or is it the nature of noun-noun and adjective-noun MWE expressions? Does the second word of the pair contribute more to the MWE complexity and thus compares better to single word targets?

## 8 Acknowledgments

I would like to express my gratitude to the anonymous reviewer for their insightful and helpful comments. I also want to thank Ragheb Al-Ghezi for proofreading and Mikko Kurimo for his comments on an earlier draft of the paper. Lastly, I want to say thank you to Science-IT team for providing computational environment used in the experiments. This work is a part of DigiTala project which is funded by the Academy of Finland (grant number 322625).

## References

- David Alfter and Ildikó Pilán. 2018. [SB@GU at the complex word identification 2018 shared task](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 315–321, New Orleans, Louisiana. Association for Computational Linguistics.
- David Alfter and Elena Volodina. 2018. [Towards single word lexical complexity prediction](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 79–88, New Orleans, Louisiana. Association for Computational Linguistics.
- Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. [Readability assessment for text simplification](#). In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9, Los Angeles, California. Association for Computational Linguistics.
- Hélène Deacon, Michael Kieffer, and Annie Laroche. 2014. [The relation between morphological awareness and reading comprehension: Evidence from mediation and longitudinal models](#). *Scientific Studies of Reading*, 18.

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.
- Lilli Kimppa, Yury Shtyrov, Suzanne Hut, Laura Hedlund, Miika Leminen, and Alina Leminen. 2019. [Acquisition of L2 morphology by adult language learners](#). *Cortex*, 116.
- Gustavo Paetzold and Lucia Specia. 2016. [SemEval 2016 task 11: Complex word identification](#). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569, San Diego, California. Association for Computational Linguistics.
- Sarah E. Petersen and Mari Ostendorf. 2007. Text simplification for language learners: A corpus analysis. In *Proceedings of Workshop on Speech and Language Technology for Education*.
- Iria del Río. 2019. [Linguistic features and proficiency classification in L2 Spanish and L2Portuguese](#). In *Proceedings of the 8th Workshop on NLP for Computer Assisted Language Learning*, pages 31–40, Turku, Finland. LiU Electronic Press.
- Matthew Shardlow. 2013. [A comparison of techniques to automatically identify complex words](#). In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. Association for Computational Linguistics.
- Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. [Complex: A new corpus for lexical complexity prediction from likert scale data](#). In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with READING Difficulties (READI)*.
- Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. [Semeval-2021 task 1: Lexical complexity prediction](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. [Luminosoin-sight/wordfreq: v2.2](#).
- Peter; Grönroos Stig-Arne; Kurimo Mikko Virpioja, Sami; Smit. 2013. [Morfessor 2.0: Python implementation and extensions for morfessor baseline](#). D4 julkaistu kehittämis- tai tutkimusraportti tai -selvitys.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google’s neural machine translation system: Bridging the gap between human and machine translation](#).
- Victoria Yaneva, Irina Temnikova, and Ruslan Mitkov. 2016. [Evaluating the readability of text simplification output for readers with cognitive disabilities](#). In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC’16)*, pages 293–299, Portorož, Slovenia. European Language Resources Association (ELRA).
- Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. [A report on the complex word identification shared task 2018](#). In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.
- Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. [CWIG3G2 - complex word identification task across three text genres and two user groups](#). In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.

# GX at SemEval-2021 Task 2: BERT with Lemma Information for MCL-WiC Task

Wanying Xie

Beijing Language and Culture University, China

xiewanying07@gmail.com

## Abstract

This paper presents the GX system for the Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC) task. The purpose of the MCL-WiC task is to tackle the challenge of capturing the polysemous nature of words without relying on a fixed sense inventory in a multilingual and cross-lingual setting. To solve the problems, we use context-specific word embeddings from BERT to eliminate the ambiguity between words in different contexts. For languages without an available training corpus, such as Chinese, we use neural machine translation model to translate the English data released by the organizers to obtain available pseudo-data. In this paper, we apply our system to the English and Chinese multilingual setting and the experimental results show that our method has certain advantages.<sup>1</sup>

## 1 Introduction

In recent years, contextual embeddings have drawn much attention. The approaches of calculating contextual embeddings include multi-prototype embeddings, sense-based and contextualized embeddings (Camacho-Collados and Pilehvar, 2018). However, it is not easy to evaluate such multiple embedding methods in one framework. Pilehvar and Camacho-Collados (2019) present a large-scale word in context dataset to focus on the dynamic semantics of words. Following and expanding them, the MCL-WiC task (Martelli et al., 2021) performs a binary classification task that indicates whether the target word is used with the same or different meanings in the same language (multilingual data set) or across different languages (cross-lingual data set). Besides, it is the first SemEval task for Word-in-Context disambiguation (Martelli et al., 2021).

<sup>1</sup>Reproducible code: <https://github.com/yingwaner/bert4wic>

A typical solution to the problems is obtaining context-specific word embeddings, such as Context2vec (Melamud et al., 2016) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019). BERT is designed to pre-train deep bidirectional representation in unlabeled texts by jointly conditioning in the left and right contexts of all layers. Due to its powerful capabilities and easy deployment, we use BERT as our major system and fine-tune on the training data released by the organizer to get the context-specific word embeddings.

In this paper, we participate in the sub-task of multilingual settings in English and Chinese. The organizer only provides English training data, and we fine-tune the pre-trained English BERT model based on this data. For Chinese tasks where no training set is available, we train a satisfactory neural machine translation (NMT) model to translate the English training set into Chinese and then fine-tune the Chinese BERT model based on the pseudo-data. The experimental results show that our method achieves 82.7% in English multilingual setting and 76.7% in Chinese multilingual setting.

## 2 Background

In this section, we will briefly introduce the word-in-context task and the structure of BERT for the sentence pair classification task.

### 2.1 Word-in-Context

The MCL-WiC task (Martelli et al., 2021) expands the Word-in-Context (WiC) (Pilehvar and Camacho-Collados, 2019) task to be multilingual and cross-lingual settings. For WiC, each instance has a target word *lemma*, which provides it with two contexts. Each context triggers the specific meaning of the word *lemma*. The task is to identify whether *lemma* in two contexts corresponds to

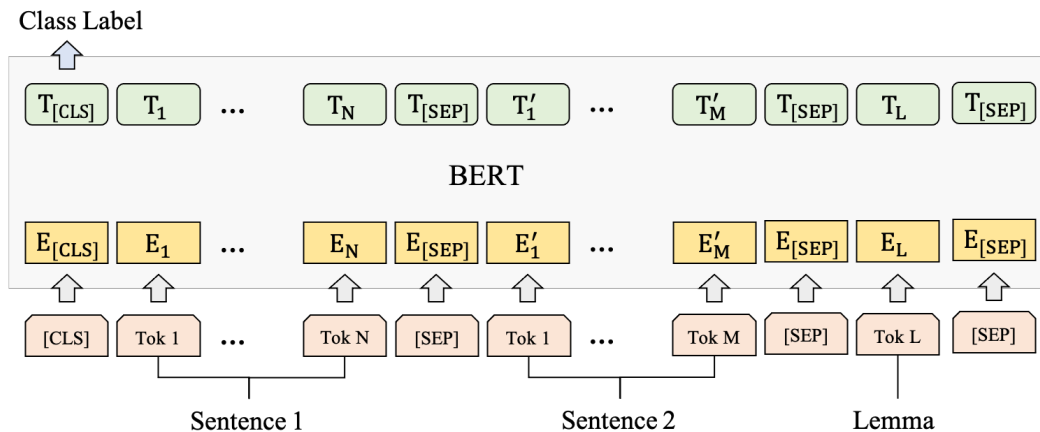


Figure 1: Overall fine-tuning procedures for our system. The token Lemma is the target word that require the system to judge whether it has the same meaning in two sentences.

the same meaning, which has been widely investigated in recent years. Wiedemann et al. (2019) perform word sense disambiguation models with contextualised representations. Hu et al. (2019) prove that the supervised derivation of time-specific sense representation is useful. Giulianelli et al. (2020) present an unsupervised approach to lexical-semantic change that makes use of contextualized word representations. Loureiro and Jorge (2019) compute sense embeddings and the relations in a lexical knowledge base. Scarlini et al. (2020) drop the need for sense-annotated corpora so as to collect contextual information for the senses in WordNet.

## 2.2 BERT

Neural contextualized lexical representation has been widely used in natural language processing, which benefits from deep learning model in optimizing tasks while learning usage dependent representations, such as ULMFiT (Howard and Ruder, 2018), ELMo (Peters et al., 2018), GPT (Radford et al., 2018, 2019), and BERT (Devlin et al., 2019). BERT is pre-trained by two unsupervised tasks: masked LM task, which is simply masking some percentage of the input tokens at random, and then predicting those masked tokens; and next sentence prediction task, which is whether the next sentence in the sentence pair is the true next sentence. In the fine-tuning phase, task-specific inputs and outputs are plugged into the BERT and all parameters are fine-tuned end-to-end.

The architecture of BERT is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al.

(2017). There are several encoder layers in the BERT model. For a single layer in Transformer encoder, it consists of a multi-head self-attention and a position-wise feed-forward network. Specifically, there are specialized input and output formats for different downstream tasks. For language pair classification task, the input format is [CLS] + Sentence 1 + [SEP] + Sentence 2 + [SEP]. At output layer, the [CLS] representation is fed into an output layer for the classification task, such as entailment, sentiment analysis, and the word-in-context disambiguation task.

## 3 System Overview

Systems proposed for both English and Chinese multilingual settings were based on BERT model (Devlin et al., 2019) with task-specific input modifications. We participate in the multilingual setting and divide the system into two parts according to the language: English setting and Chinese setting.

### 3.1 English Setting

Following Devlin et al. (2019), we initialize our model with the well pre-trained model, which has been trained on the large-scale data set and obtained the general knowledge. Then we fine-tune the model on the English parallel sentences released by the organizers.

**Model Architecture** The model architecture in the fine-tuning stage is shown in Figure 1. On the basis of the original BERT input, *lemma* token is added, which is the target word that needs the system to judge whether it has the same meaning

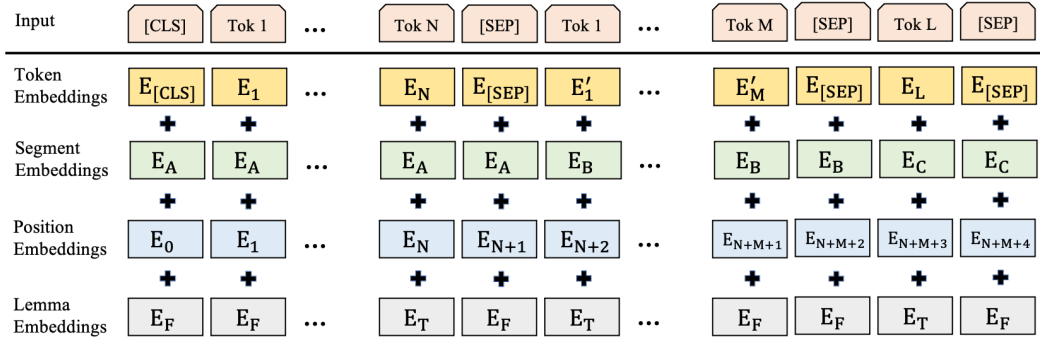


Figure 2: Our system input representation. The input embeddings are the sum of the token embeddings, the segment embeddings, the position embeddings, and the lemma embeddings. It is only at the position of the lemma tokens that the lemma embedding is  $E_T$ . In this example, we assume that Tok N in sentence 1 and Tok 1 in sentence 2 are also lemma tokens.

in the sentence pair. For instance, given the sentence pairs 'They opposite to the policies designed to tackle inflation.' and 'I tackled him about his heresies.', the standard input format is:

[CLS] They opposite to the policies designed to tackle inflation . [SEP] I tackled him about his heresies . [SEP] tackle [SEP] where *tackle* is the lemma token, which is also the word that needs to be judged by the system whether it has the same meaning in two sentences. In this way, we emphasize the target word so that the output  $T_{[CLS]}$  of the output layer can express whether the lemma token is synonymous in the two sentences.

**Input Representation** In addition, we also made some modifications for the input representation, which is made up of the sum of the corresponding token, segment, and position embeddings according to Devlin et al. (2019). The input representation of our system is shown in Figure 2. We adjust the segment embeddings of the lemma token to further emphasize the importance of the target word in the whole sentence pair, which is represented as  $E_C$ . Moreover, we introduce lemma embeddings in the input representation. Lemma embeddings are similar to segment embeddings, but segment embeddings are to distinguish between sentence 1 and sentence 2, while lemma embeddings are to distinguish between the position of lemma tokens and the position of other tokens. Only the lemmas in sentence 1 and sentence 2 and the final lemma Tok L will be marked  $E_T$ , and the other positions will be marked  $E_F$ , that is, for a training example, there will be three lemma markers  $E_T$  in lemma embeddings. In this way, we enhance the relationship of lemma tokens to make them

more closely connected, and at the same time highlight and emphasize the position and importance of lemma tokens, so that the final output can obtain enough lemma token information.

### 3.2 Chinese Setting

The multilingual setting in Chinese is more difficult because there is no available training data in Chinese, so it is not possible to fine-tune the pre-trained BERT model. In order to solve this problem, we introduce neural machine translation method.

**Neural Machine Translation** Due to the superior performance of Transformer, we use it as our neural machine translation model. We first train an English-to-Chinese translation model on an open-source dataset and evaluate its performance to ensure that it has sufficient translation quality. Then, we use this translation model to translate sentence 1 and sentence 2 from the English training set released by the organizer into Chinese, respectively, and regard the generated sentences as the training data of Chinese MCL-WiC task. Finally, we use the pre-trained Chinese BERT model to fine-tune this generated data set to get our final model.

**Model Architecture** The system in Chinese setting is somewhat different from the system in English setting in that there is no lemma token. We use machine translation to translate English training data into Chinese, because every token in a sentence has a context, so sentence to sentence translation does not change the meaning of the whole sentence much. However, a lemma token has no context, so it is difficult for translation model to choose which token to translate into the target



	English				Chinese			
	Valid	$\Delta$	Test	$\Delta$	Valid	$\Delta$	Test	$\Delta$
Fine-tuning	81.2	-	81.6	-	67.9	-	76.7	-
+Lemma Token	81.8	0.6	82.1	0.5	67.1	-0.8	76.0	-0.7
+Segment $E_C$	82.2	0.4	82.4	0.3	67.3	0.2	76.3	0.3
+Lemma Embeddings	82.5	0.3	82.7	0.3	-	-	-	-

Table 1: Main results on English and Chinese tasks. The measure is accuracy (%). The '+' in systems represent an increase in modules of the system in the previous row.  $\Delta$  represents the difference between the result of the current system and that of the previous row.

language, because it may correspond to multiple meanings. Therefore, the final submitted system to the task has no lemma token, no segment embedding  $E_C$  and no lemma embeddings. However, in order to analyze the role of lemma tokens in this multilingual setting, we will report the results with lemma token and segment embedding in Table 1 and Section 5.1. In this case, the lemma token will be translated to the most common Chinese word.

## 4 Experimental Setup

In this section, we will describe the experimental settings for English and Chinese in detail.

### 4.1 English Setting

Take one pre-trained English cased BERT *base* model<sup>2</sup> with 12 layer, 768 hidden, 12 heads, and 110M parameters, and fine-tune on the English training data in 5 epochs with batch size is 16 and max sequence length is 128. The dropout rate is 0.2 and other settings are followed Devlin et al. (2019).

### 4.2 Chinese Setting

The fine-tuning setups are the same as the English ones, except that the pre-training model is a Chinese BERT *base*<sup>3</sup> with a layer of 12, hidden size of 768, heads of 12, and parameters of 110M.

For machine translation model, we implement Transformer *base* model (Vaswani et al., 2017) using the open-source toolkit *Fairseq-py* (Ott et al., 2019). The training data of English-Chinese is from UNPC v1.0 and MultiUN v1 in WMT17<sup>4</sup>, which are total 30.4M sentences. We trained the model with dropout = 0.1 and using Adam optimizer (Kingma and Ba, 2015) with  $\beta_1 = 0.9$ ,

<sup>2</sup>[https://storage.googleapis.com/bert\\_models/2018\\_10\\_18/cased\\_L-12\\_H-768\\_A-12.zip](https://storage.googleapis.com/bert_models/2018_10_18/cased_L-12_H-768_A-12.zip)

<sup>3</sup>[https://storage.googleapis.com/bert\\_models/2018\\_11\\_03/chinese\\_L-12\\_H-768\\_A-12.zip](https://storage.googleapis.com/bert_models/2018_11_03/chinese_L-12_H-768_A-12.zip)

<sup>4</sup><http://www.statmt.org/wmt17/translation-task.html>

$\beta_2 = 0.98$ , and  $\epsilon = 10^{-9}$ . The translation is detokenized and then the quality is 35.0, which is evaluated using the 4-gram case-sensitive BLEU (Papineni et al., 2002) with the *SacreBLEU* tool (Post, 2018).<sup>5</sup> This translation model achieves satisfactory results, which shows that the method of translating English training set into Chinese has theoretical basis and feasibility.

## 5 Results

In this section, we will first report the main results of English and Chinese multilingual setting and analyze the importance of all the factors in the system. Then, we explore the probability of error for each part of speech.

### 5.1 Main Results

We conduct our experiments based on BERT frame<sup>6</sup>. The specific meanings of each system in the main experiment are as follows:

**Fine-tuning** Following the standard fine-tuning format of Devlin et al. (2019), sentences 1 and 2 are connected by [SEP].

**+Lemma Token** Lemma token is added on the basis of the previous system, and the training input at this time is '[CLS] + Sentence 1 + [SEP] + Sentence 2 + [SEP] + Lemma + [SEP]'.

**+Segment  $E_C$**  Based on the previous system, the segment embedding of the final lemma token is set to  $E_C$ .

**+Lemma Embeddings** Lemma embedding will be added on the basis of the previous system, and the input representation at this time consists of four parts: token, segment, position, and lemma embeddings.

The main results are shown in Table 1, and the specific analysis is as follows:

<sup>5</sup>BBLEU+case.mixed+lang.en-zh+numrefs.1+smooth.exp+test.wmt17+tok.zh+version.1.4.4

<sup>6</sup><https://github.com/google-research/bert>

	English		Chinese	
	Num	Acc	Num	Acc
NOUN	528	83.14	554	78.70
VERB	298	83.22	364	75.82
ADJ	144	81.94	62	69.35
ADV	30	73.33	20	60.00
Overall	1000	82.70	1000	76.70

Table 2: Error analysis on the test set. Num represents the number of examples of this part of speech in the test set, and Acc represents the accuracy (%) of the current part of speech.

**English** Fine-tuning alone can obtain relatively good performance, and the performance has been further improved with the introduction of the three new modules. Here we conduct experiments to check their influence on our method by adding them one by one. With the addition of +Lemma Token, our system has a significant improvement, while the improvement brought by the other two modules is slightly lower, indicating that the presence or absence of lemma token has a greater impact.

**Chinese** Fine-tuning achieves the best performance in this task. After the introduction of +Lemma Token, the result shows that the effect is greatly reduced, which may be because the translated lemma token is not necessarily the appropriate translation result. Because fine-grained translations of individual tokens have no context, they often fail to translate properly, as mentioned in Section 3.2. However, after the introduction of +Segment  $E_C$ , the effect has been slightly improved, which proves that our idea is effective. Because it is difficult to get the exact position of the lemma word after translation, there is no result of +Lemma Embeddings. Based on the above results, we use Fine-tuning as the final system for the Chinese task. In other words, our final model has no lemma token, no segment embedding  $E_C$  and no lemma embeddings.

## 5.2 Error Analysis

Lemma tokens have different parts of speech, so we think about the relationship between the accuracy of system prediction and the parts of speech of lemma tokens. Based on this, we reported the accuracy of each part of speech in the test set, as shown in Figure 2.

There are similar findings of error analysis for English and Chinese tasks. The order of the data

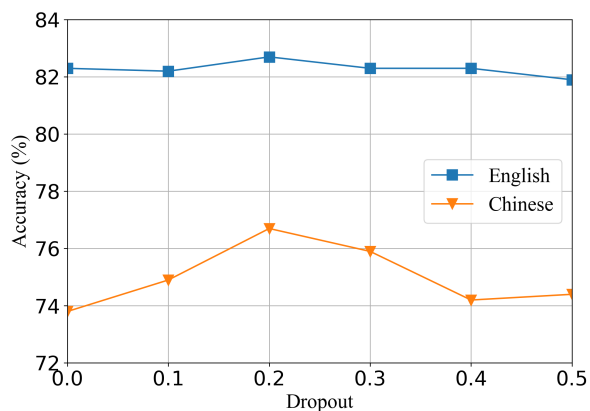


Figure 3: The performance of different dropouts on test set in English and Chinese tasks.

volume on the test set is NOUN, VERB, ADJ, and ADV. The distribution of these data types is consistent with the training set, and each part of speech in the training set is 4123, 2269, 1429, and 175, which is also in descending order and occupies roughly the same proportion of each part of speech. Therefore, the parts of speech with more data in the training set often get better performance on the test set, which indicates the importance of data size. More data can enable the model to learn more classification knowledge and behavior, which affects the prediction results.

## 5.3 Impact of Dropout

To analyze the importance of dropout, we conducted experiments by using different dropouts on both English and Chinese test sets, and the results are shown in Figure 3. As we can see, the performance of the two tasks increases with the increment of dropout rate and reach the best performance when dropout rate equals 0.2. As dropout rate continues to increase, the performance deteriorates, which indicates that too many lost parameters may make the model difficult to converge.

Besides, Dropout performs differently in terms of data quality. In general, real corpus (English) should be of better quality than pseudo corpus (Chinese). On this basis, the performance of different dropout models on high-quality real corpus is relatively stable, with a gap of less than 1%, while the performance of pseudo corpus fluctuates greatly, with a gap of 3%.

## 6 Conclusion

In this paper, we describe the GX system participating in the MCL-WiC task. In order to obtain the

general basic knowledge, we use the pre-trained BERT model and then fine-tune it on the data released by the organizer. In order to further emphasize the relationship between sentence pairs and the importance of lemma, we introduce three new factors: lemma token, lemma segment embedding, and lemma embedding, and finally get better results. Our system reaches 82.7% in English multilingual setting and 76.7% in Chinese multilingual setting.

## Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. We thank Shuhao Gu for his helpful suggestions and inspiration.

## References

- José Camacho-Collados and Mohammad Taher Pilehvar. 2018. [From word to sense embeddings: A survey on vector representations of meaning](#). *J. Artif. Intell. Res.*, 63:743–788.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Mario Giulianelli, Marco Del Tredici, and Raquel Fernández. 2020. [Analysing lexical semantic change with contextualised word representations](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 3960–3973. Association for Computational Linguistics.
- Jeremy Howard and Sebastian Ruder. 2018. [Universal language model fine-tuning for text classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339. Association for Computational Linguistics.
- Renfen Hu, Shen Li, and Shichen Liang. 2019. [Diachronic sense modeling with deep contextualized word embeddings: An ecological view](#). In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3899–3908. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Daniel Loureiro and Alípio Jorge. 2019. [Language modelling makes sense: Propagating representations through wordnet for full-coverage word sense disambiguation](#). In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5682–5691. Association for Computational Linguistics.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. [context2vec: Learning generic context embedding with bidirectional LSTM](#). In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 51–61. ACL.
- Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. [Bleu: a method for automatic evaluation of machine translation](#). In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and José Camacho-Collados. 2019. [Wic: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 1267–1273. Association for Computational Linguistics.
- Matt Post. 2018. [A call for clarity in reporting BLEU scores](#). In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. [SenseBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation](#). In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8758–8765. AAAI Press.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.
- Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. [Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings](#). In *Proceedings of the 15th Conference on Natural Language Processing, KONVENS 2019, Erlangen, Germany, October 9-11, 2019*.

# PALI at SemEval-2021 Task 2: Fine-Tune XLM-RoBERTa for Word in Context Disambiguation

Shuyi Xie, Jian Ma, Haiqin Yang<sup>§</sup>, Lianxin Jiang, Yang Mo, and Jianping Shen

Ping An Life Insurance, Ltd.

Shenzhen, Guangdong province, China

{XIESHUYI542, MAJIAN446, JIANGLIANXIN769, MOYANG853, SHENJIANPING324}@pingan.com.cn

<sup>§</sup> the corresponding author, email: hqyang@ieee.org

## Abstract

This paper presents the PALI team’s winning system for SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation. We fine-tune XLM-RoBERTa model to solve the task of word in context disambiguation, i.e., to determine whether the target word in the two contexts contains the same meaning or not. In implementation, we first specifically design an input tag to emphasize the target word in the contexts. Second, we construct a new vector on the fine-tuned embeddings from XLM-RoBERTa and feed it to a fully-connected network to output the probability of whether the target word in the context has the same meaning or not. The new vector is attained by concatenating the embedding of the [CLS] token and the embeddings of the target word in the contexts. In training, we explore several tricks, such as the Ranger optimizer, data augmentation, and adversarial training, to improve the model prediction. Consequently, we attain the first place in all four cross-lingual tasks.

## 1 Introduction

This year, the SemEval-2021 task 2, multilingual and cross-lingual word-in-context (WIC) disambiguation (Martelli et al., 2021), defines the task of identifying the polysemous nature of words without relying on a fixed sense inventory in a multilingual and cross-lingual setting. The task aims to perform a binary classification task to determine whether the target word contains the same meaning or not in two given contexts under both the same language (multilingual) setting and the different languages (cross-lingual) setting. In the multilingual setting, the tasks consist of English-English (En-En), Arabic-Arabic (Ar-Ar), French-French (Fr-Fr), Chinese-Chinese (Zh-Zh) and Russian-Russian (Ru-Ru) while in the cross-lingual setting, the tasks consist of English-Chinese (En-Zh),

English-French (En-Fr), English-Russian (En-Ru), and English-Arabic (En-Ar).

The tasks contain the following challenges:

- The same word may deliver different meanings in different context (Lei et al., 2021).
- The training data is scarce. For example, in the multilingual tasks, there is only training data in En-En, while in the cross-lingual tasks, there is no training data.

To overcome these challenges, we explore the uniqueness of the tasks and implement several key technologies:

- First, we follow (Botha et al., 2020) to specially design an input tag for the multilingual pre-training XLM-RoBERTa model to emphasize the target word in the contexts. That is, the target word is encompassed by the special symbols of `<t>` and `</t>`. Meanwhile, the given two contexts are concatenated by the `<SEP>` token.
- Second, we apply data augmentation and add external data from WordNet to enrich the training data. It is noted that we only expand the data in the task of En-En and do not consider other techniques, e.g., back-translation, for the cross-lingual tasks. Adversarial training is also applied to learn more robust embeddings for target words. The Ranger optimizer with the look-ahead mechanism in the AdamW optimizer is adopted to speed up the convergence of training.
- Finally, we construct a new vector on the fine-tuned embeddings, i.e., concatenating the embedding of the [CLS] token and the learned embeddings of the target words’ in both contexts. The new vector is then fed into a fully-connected network to produce the binary classification prediction. Cross-validation and model ensemble are also applied to attain a robust output.

The rest of this paper is organized as follows: In Sec. 2, we briefly introduce related work. In Sec. 3, we detail our proposed system. In Sec. 4, we present the experimental setup, procedure, and the results. Finally, we conclude our work in Sec. 5.

## 2 Related Work

The SemEval-2021 task 2 aims to handling the tasks of multilingual and cross-lingual word-in-context disambiguation (Martelli et al., 2021), i.e., to determine whether the target word contains the same meaning in both given contexts. In the following, we will elaborate several related work.

Some recent effort, e.g., (Pilehvar and Camacho-Collados, 2018), has been conducted to curate and release datasets to solve the task of WiC disambiguation. Though it can be narrowed down to binary classification, some techniques have to be implemented to enhance the model performance. For example, the trick of input highlighting mechanism (Botha et al., 2020) can be facilitated to promote the importance of the target word. The idea of unifying entity linking and word sense disambiguation (Moro et al., 2014) can be borrowed to solve the task. The idea of freezing the trained model for other languages (Artetxe et al., 2020) can be explored to relieve the issue of no training data in the cross-lingual tasks.

Recently, due to the superior performance in tackling NLP tasks (Yang et al., 2021; Yang and Shen, 2021; Wang et al., 2021), pre-trained language models, such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), start to dominate the way of word representations than static word embedding methods, e.g., Word2Vec (Mikolov et al., 2013) and FastText (Joulin et al., 2017). Especially, the XLM-RoBERTa (Conneau et al., 2020) model is a newly released large cross-lingual language model based on RoBERTa and is trained on 2.5TB filtered CommonCrawl data in 100 languages. Different from other XLM models, XLM-RoBERTa does not require the language token to understand which language is used and can determine the correct language from the input ids. It is a powerful tool for understanding multilingual languages and is very helpful for solving the WiC disambiguation task under the cross-lingual setting. Hence, we choose XLM-RoBERTa in our system.

A critical issue of the task is lack of training data. Though existing methods, e.g., lexical substitution (Zhang et al., 2015), back translation (Xie

et al., 2020), and data augmentation (Fadaee et al., 2017), can be applied to enrich the data, we mainly explore the usage of WordNet (Fellbaum, 1998) and the technique of pseudo labelling (Wu and Prasad, 2018) because WordNet contains rich synonyms while pseudo labelling is effective to utilize the abundant unlabeled data via their pseudo labels.

Adversarial training (Tramèr et al., 2018) is an effective method to regularize parameters by introducing noise and to improve model robustness and generalization. We also explore its possibility in fine-tuning XLM-RoBERTa to increase the robustness of the learned the word embeddings.

## 3 Overview

In the following, we present the task definition, data preprocessing, and our proposed system design.

### 3.1 Task Definition

The task of WiC disambiguation is framed by a binary classification task. Each instance in WiC has a target word  $w$ , whose part-of-speech is in {NOUN, VERB, ADJ, ADV}, with two given contexts,  $c_1$  and  $c_2$ . Each of these contexts triggers a specific meaning of  $w$ . The task is to identify if the occurrences of  $w$  in  $c_1$  and  $c_2$  correspond to the same meaning or not. Figure 1 illustrates an example from the dataset.

```
{
  "target word": "play",
  "sentence1": "In that
    context of coordination
    and integration, Bolivia
    holds a key play in any
    process of infrastructure
    development.",
  "sentence2": "In schools,
    when water is needed, it
    is girls who are sent to
    fetch it, taking time away
    from their studies and
    play."
}
```

Figure 1: An example from the WiC disambiguation task.

### 3.2 Data Preprocessing

The training dataset consists of two files in the JSON format: the .data file and the .gold file. The .data file contains the following information:

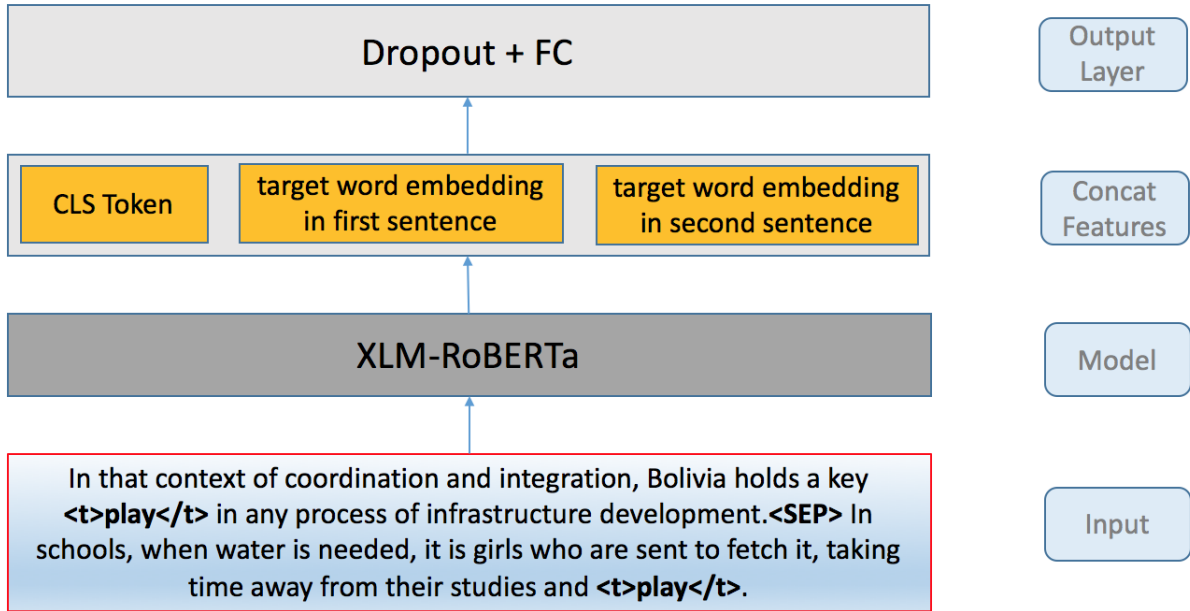


Figure 2: Fine-tuned XLM-RoBERTa model architecture.

	Training	Test
No. of target words	3,726	491
No. of pairs	8,000	1,000
Min. tokens	6	5
Avg. tokens (original)	24	26
Max. tokens (original)	88	116
Max. tokens (post-proc.)	81	81

Table 1: Statistics of the data

unique id of the pair, target lemma, part-of-speech in {NOUN, VERB, ADJ, ADV}, the first sentence, the second sentence, the start and the end indices (zero-based numbering) of the target word in the first and the second context, respectively. The .gold file contains unique id of the pair and the label, which is represented by T or F.

For the training dataset, we clean up the text by completing word abbreviation, removing special punctuation, and segmenting the sentences into subword lists by Byte-Pair Encoding (BPE) (Sennrich et al., 2015). Since it is difficult to capture the meaning of the target word in the context for long sentences (Pan et al., 2019; Zhu et al., 2021), we limit the length of each sentence with maximum 40 words before and after the target word.

We include additional resource, WordNet, to augment our training data because WordNet is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive

synonyms (synsets), each expressing a distinct concepts. Here, we randomly select example sentences of target word in WordNet to expand our training corpus, which increases around 30% of training data. By such preprocessing, we obtain the dataset and report the statistics in Table 1.

### 3.3 Model Design

Figure 2 outlines our model architecture, which consists of four modules, i.e., input design, model learning, final feature construction, and the classifier. The whole framework is based on fine-tuning the pre-trained XLM-RoBERTa model to conduct binary classification on two given contexts. Different from the inputs for XLM-RoBERTa, the input of our system contains of the following modifications: first, in order to highlight the target word in the contexts, we borrow the setting in (Lei et al., 2017; Botha et al., 2020) by adding special symbols <t> and </t> to embrace the target word in the contexts. Given the example presented in Fig. 1, the target word of “play” is then embraced by the additional symbols, <t> and </t> in the contexts. Second, we concatenate the given two contexts by <SEP>. Figure 2 illustrates the result in the input module. Moreover, in the experiment, we exchange the order of the contexts to get more training data.

After learning the tokens’ representations by XLM-RoBERTa, we construct a new vector by concatenating the [CLS] token’s representation in the last layer of XLM-RoBERTa and the representa-

tions of the target word in both sentences. As BPE tokenization may separate a target word into several subwords, we compute its representation by averaging the corresponding representations. Next, the newly constructed feature is fed into a fully-connected network to compute the final binary prediction probability.

During training, we conduct the following techniques to increase the model convergence and robustness:

- **Optimizer.** We adopt the Ranger (Yong et al., 2020) optimizer to replace the AdamW because it is a more synergistic optimizer combining rectified Adam and the look-ahead mechanism with gradient centralization in one optimizer.
- **Adversarial training.** We apply the fast gradient method (Miyato et al., 2017) in the training to obtain more stable word representations.
- **Cross validation.** We also apply stratified  $K$ -fold cross validation on the training set and the development set. For each fold, we hold the group as a local test set and set the remaining groups as the training set. We then average the model prediction on each fold as the final prediction to obtain more robust results.
- **Pseudo labelling.** Pseudo labelling (Wu and Prasad, 2018) is an effective semi-supervised learning method to utilize the abundant unlabeled data via their pseudo labels. In this work, we first train our model on the training set. Next, we apply the trained model to predict the En-En multi-lingual test set and use the predicted labels as our pseudo labels. Finally, both the training set and the En-En pseudo labels are included to train a final model. Especially, we observe that by this trick, this final model can improve the prediction performance on cross-lingual tasks slightly.

It is noted that in the cross-lingual tasks, we do not back-translate the subwords to English but apply the same model trained from the En-En dataset because it allows us to maintain the target word in the corresponding languages seamlessly. This is similar to the procedure in (Artetxe et al., 2020).

## 4 Experiments

In the following, we detail our experimental setup and present the results with analysis.

### 4.1 Setup

Our code is written in Pytorch based on the Huggingface Transformer library<sup>1</sup> for XLM-RoBERTa. Other hyperparameters are set based on our hand-on experience. For example, the seed for the random generator is set to 3,999. The batch size is set to 10 and the hidden feature size is 1,024. The maximum length limit of a context is 240 though it is unreachable because we have conducted trimming in the data preprocessing procedure. The dropout rate is tested from {0.2, 0.3, 0.25, 0.28} and finally fixed to 0.28.  $K$  in the stratified cross validation is set to 5. The two special tokens,  $\langle t \rangle$  and  $\langle /t \rangle$ , are included into the word dictionary for learning.

The training data consists of the official En-En multilingual training corpus and the contexts from WordNet. At the beginning, we choose XLM-RoBERTa<sub>Base</sub> as the backbone of our system to explore the possibility of our implementation tricks. After identifying the effectiveness of the designed input in Sec. 3.2, we apply XLM-RoBERTa<sub>Large</sub> to tune the corresponding hyperparameters, such as changing the learning rate, the batch size, the dropout rate, and the early stop mechanism. Furthermore, we observe that long contexts may ignore the importance of the target word in the contexts. Hence, we center on the target words to cut off the contexts at both ends with a certain length. To further strengthen the influence of the target word in a context, we concatenate the embedding of the [CLS] token with the embeddings of the target word in the contexts as the final input for the logit fully-connected network. From our experiment, this strategy can significantly boost the model performance while improving the convergence.

### 4.2 Results

Table 2 reports the results of different implementation strategies on the tasks. From the results, we observe that

- By replacing XLM-RoBERTa<sub>Base</sub> with XLM-RoBERTa<sub>Large</sub>, we can gain at least 3% improvement on all tasks.
- By applying Ranger optimizer, we attain the results in Large+RO, which gain an average increase of 0.2% per task. We conjecture the improvement comes from the fact that the model converges to a more optimal solution.
- In Large+RO+LRA, we vary the learning rate

<sup>1</sup><https://github.com/huggingface/transformers>



Strategy	Avg	En-En	Fr-Fr	Ru-Ru	Zh-Zh	Ar-Ar	En-Ru	En-Zh	En-Fr	En-Ar
Base	80.8	85.5	80.7	78.7	80.9	79.1	81.0	81.9	79.1	80.4
Large	85.1	88.2	84.2	84.3	87.0	82.6	84.6	85.7	85.6	83.9
Large + RO	85.4	88.7	85.3	85.1	86.9	83.3	84.7	85.9	84.7	83.6
Large + RO + LRA	85.4	89.2	84.9	85.1	86.8	83.2	84.8	85.6	85.2	84.1
Large + RO + LRA + ES	85.5	89.0	84.8	85.5	86.9	83.1	85.2	85.2	85.4	84.1
Large + RO + CTWE	86.3	90.0	85.8	85.9	87.1	83.9	86.0	86.1	85.4	86.2
Large + RO + CTWE + HC	86.3	89.9	85.7	85.9	87.2	84.2	85.6	86.7	85.0	86.3
Large + RO + CTWE + HC + WordNet	86.5	91.6	85.8	85.7	87.1	84.0	85.3	87.1	85.6	86.0
Large + RO + CTWE + HC + WordNet + AT	87.0	91.1	86.3	85.9	87.9	85.1	86.3	87.2	86.3	86.9
Large + RO + CTWE + HC + WordNet + AT + PL	<b>88.1</b>	<b>91.7</b>	<b>86.9</b>	<b>86.5</b>	<b>89.2</b>	<b>86.5</b>	<b>88.0</b>	<b>87.9</b>	<b>88.6</b>	<b>87.2</b>

Table 2: Results of fine-tuning XLM-RoBERTa under different strategies. The abbreviation is defined as follows: Base: XLM-RoBERTa<sub>Base</sub>; Large: XLM-RoBERTa<sub>Large</sub>; RO: Ranger Optimizer; LRA: learning rate adjustment; ES: early stop; CTWE: concatenating target words’ embeddings; HC: the best parameters for LRA and ES; AT: adversarial training; PL: pseudo labels.

from  $1.5e-5$ ,  $1.3e-5$ ,  $1.2e-5$ , to  $1.21e-5$  progressively and finally find that when the learning rate is  $1.2e-5$ , we attain the best performance. We then search the optimal epoch for the early stop by setting the maximum number of epoch to 10. In Large+RO+LRA+ES, we observe the optimal epoch for early stop (the patience value) is 3. These parameters are then fixed for HC. From the results, we notice that tuning the learning rate and adopting the early stop mechanism can improve the model performance accordingly.

- By concatenating target the word embedding, we obtain the results in Large+RO+CTWE, and actually, our model can be trained with fewer epochs and attain around 1.1% improvement on average.
- By adding more training data from WordNet, we get another 0.2% average improvement in Large+RO+CTWE+HC+WordNet. We conjecture the improvement mainly comes from the increase of the training data.
- By adding the Pseudo label data, we can gain another 0.8% average improvement. The score of EN-EN test dataset is generally higher than other test dataset. We discover that the first 462 pieces of English test dataset have the same target word as test dataset in other tasks. Therefore, adding EN-EN pseudo label helps predict other tasks.

In sum, we conclude that by applying XLM-

RoBERTa<sub>Large</sub> on the Ranger optimizer, the target word embedding concatenation mechanism, more external training data, and pseudo labels, we can improve the model performance accordingly.

Finally, our system attains the champion on the En-Ar, En-Fr, En-Ru, and En-Zh cross-lingual tasks. In multilingual tasks, we also sit at eighth place, seventh place, sixth place, seventh place, and fifth place for the En-En, Ar-Ar, Fr-Fr, Ru-Ru, Zh-Zh tasks, respectively.

## 5 Conclusion

In this paper, we present our system to tackle the word-in-context disambiguation task. We fine-tune the XLM-RoBERTa model to solve both multilingual and cross-lingual word-in-context disambiguation tasks. We specifically design the input format to emphasize the target word in two contexts and promote the importance of the target word by concatenating the embeddings in the corresponding context with the [CLS] token to output the classification probability. We apply several training tricks to improve the robustness of model and attain improvement during this procedure. The competition results demonstrate the effectiveness of our implementation. In the future, we plan to explore more model architecture to boost the performance for multilingual tasks.

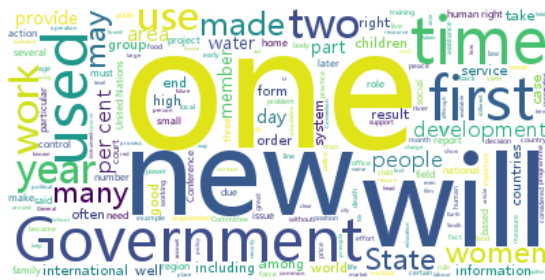
## References

- Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. [On the cross-lingual transferability of monolingual representations](#). In *ACL 2020*, pages 4623–4637. Association for Computational Linguistics.
- Jan A. Botha, Zifei Shan, and Daniel Gillick. 2020. [Entity linking in 100 languages](#). In *EMNLP*, pages 7833–7845. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *ACL*, pages 8440–8451. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.
- Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. [Data augmentation for low-resource neural machine translation](#). *CoRR*, abs/1705.00440.
- Christiane Fellbaum. 1998. Wordnet: An electronic lexical database.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomáš Mikolov. 2017. [Bag of tricks for efficient text classification](#). In *EACL*, pages 427–431. Association for Computational Linguistics.
- Wenqiang Lei, Yisong Miao, Runpeng Xie, Bonnie Webber, Meichun Liu, Tat-Seng Chua, and Nancy F Chen. 2021. [Have we solved the hard problem? it’s not easy! contextual lexical contrast as a means to probe neural coherence](#). In *AAAI*.
- Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. [Swim: a simple word interaction model for implicit discourse relation recognition](#). In *IJCAI*, pages 4026–4032.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. [SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation \(MCL-WiC\)](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#). In *ICLR*.
- Takeru Miyato, Andrew M. Dai, and Ian J. Goodfellow. 2017. [Adversarial training methods for semi-supervised text classification](#). In *ICLR*. OpenReview.net.
- Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. [Entity linking meets word sense disambiguation: a unified approach](#). *Trans. Assoc. Comput. Linguistics*, 2:231–244.
- Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. [Recent advances in neural question generation](#). *arXiv preprint arXiv:1905.08949*.
- Mohammad Taher Pilehvar and José Camacho-Collados. 2018. [Wic: 10,000 example pairs for evaluating context-sensitive representations](#). *CoRR*, abs/1808.09121.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. [Neural machine translation of rare words with subword units](#). *CoRR*, abs/1508.07909.
- Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. 2018. [Ensemble adversarial training: Attacks and defenses](#). In *ICLR*. OpenReview.net.
- Xinyi Wang, Haiqin Yang, Liang Zhao, Yang Mo, and Jianping Shen. 2021. [Refbert: Compressing bert by referencing to pre-computed representations](#). In *IJCNN*.
- Hao Wu and Saurabh Prasad. 2018. [Semi-supervised deep learning using pseudo labels for hyperspectral image classification](#). *IEEE Trans. Image Process.*, 27(3):1259–1270.
- Qizhe Xie, Zihang Dai, Eduard H. Hovy, Thang Luong, and Quoc Le. 2020. [Unsupervised data augmentation for consistency training](#). In *NeurIPS*.
- Haiqin Yang and Jianping Shen. 2021. [Emotion dynamics modeling via bert](#). In *IJCNN*.
- Haiqin Yang, Xiaoyuan Yao, Yiqun Duan, Jianping Shen, Jie Zhong, and Kun Zhang. 2021. [Progressive open-domain response generation with multiple controllable attributes](#). In *IJCAI*.
- Hongwei Yong, Jianqiang Huang, Xiansheng Hua, and Lei Zhang. 2020. [Gradient centralization: A new optimization technique for deep neural networks](#). In *ECCV*, volume 12346 of *Lecture Notes in Computer Science*, pages 635–652. Springer.
- Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. [Character-level convolutional networks for text classification](#). In *NIPS*, pages 649–657.
- Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. [Retrieving and reading: A comprehensive survey on open-domain question answering](#). *arXiv preprint arXiv:2101.00774*.





(a) The validation set data



(b) The test set data

Figure 2: The word cloud diagram of the validation set and test set data provided by the task organizer team. The result shown in the figure is the data after removing the stop words.

former Encoder (Vaswani et al., 2017) achieved the best results in many NLP tasks.

We participated in SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC) English task. This task is to predict whether a word with the same part of speech has the same meaning in a sentence pair (Martelli et al., 2021). We are inspired by the work of Chen, Weilong and others on the task of predicting the influence of context on word similarity (Chen et al., 2020), and use methods based on RoBERTa (Liu et al., 2019) and Tf-Idf (Ramos et al., 2003) to complete the task. At the same time, we also tried to combine ALBERT (Lan et al., 2020) with BERT (Devlin et al., 2019) and Tf-Idf to observe their performance on the English data set. We introduce our methods and experiments in detail in Sections 2 and 3. Our model code can provide reference <sup>1</sup>.

## 2 Data and Methods

In this section, we will introduce the data we use in the task and the models and methods we use.

### 2.1 Data Description

The task organizer team provides each team with training data sets, validation data sets, and test data sets related to the "Multilingual and Cross-lingual Word-in-Context Disambiguation" task. Because we only successfully submitted the test set prediction results of the English task, we only discuss the English data set here. The training data set and the validation data set are composed of two parts. The first part contains the ID, the lemma of the target word, the part of speech of the target word, the sentence pair data, and the position index of the target word in the sentence pair. The target word is usually only one word, and they have the same part

<sup>1</sup><https://github.com/Hub-Lucas/hub-at-task2>

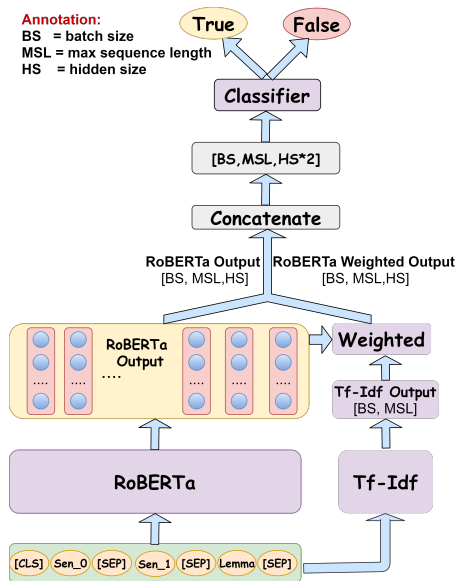


Figure 3: The model structure and data flow we used in the task.

of speech in the sentence pair. The second part is whether the target words appearing in the sentence pair are tags with the same meaning.

If two words have the same meaning, it is "True", otherwise it is "False". The sentence lengths in the sentence pairs are not the same. Compared with the training data set and the validation data set, the test set only contains the first part mentioned above. We need to use our method to predict whether the same words appearing in sentence pairs in the test set have the same meaning. Table 1 shows a sample of sentence pair data we used in the task.

There are 8000 and 1000 data in the training set and validation set respectively. The proportions of the "True" label and the "False" label in the training set and the validation set are the same, both are 50% and 50%. There are 1000 pieces of data in the test set. Information about word frequency

ID	Lemma	Part of Speech	Sentence	Start	End
151	excess	NOUN	We want to rebuild our country, which was dismantled by the <b>excesses</b> of Mobutu	60	68
151	excess	NOUN	More often than not, words per page are well in <b>excess</b> of that standard.	48	54

Table 1: The sample data of a pair of sentence pairs we use in the task.

$$[Tf - Idf\_Output]_i \times [RoBERTa\_Output]_i = [Weighted]_i \quad (1)$$

$$[Tf - Idf\_Output]_i^T \times [Weighted]_i = [RoBERTa\_Weighted\_Output]_i \quad (2)$$

$$0 \leq i < batch\_size \quad (3)$$

will be involved in our method. We use word cloud graphs to visualize the text data in the training set and the text data in the test set. The word cloud image clearly shows us the characteristics of word frequency distribution in the text data set. Figure 1 and Figure 2 show the word frequency information in the training set, validation set, and test set.

## 2.2 Methods

Combined with the analysis and understanding of task description and task data set, we chose to develop a system based on RoBERTa and Tf-Idf. Besides, we also tried to use the combination of ALBERT (Lan et al., 2020), BERT (Devlin et al., 2019) and Tf-Idf to verify their effect on the verification set. Due to the addition of the attention mechanism, Transformer has achieved good results in multi-tasking in the field of natural language. The three models of BERT, ALBERT, and RoBERTa are all based on the improvement of the transformer architecture. Compared with BERT, ALBERT not only has fewer parameters, but also has the characteristics of parameter sharing between different layers (Lan et al., 2020; Devlin et al., 2019). Therefore, ALBERT is better than BERT in terms of memory space and training time. Compared with ALBERT (Lan et al., 2020), RoBERTa (Liu et al., 2019) does not perform the task of predicting the next sentence during the pre-training process, and also uses a new dynamic masking mechanism. At the same time, the pre-training time of the RoBERTa model is longer, using a larger batch size, and the corpus data used for pre-training is also larger (Liu et al., 2019).

In our system, the first step is to use the pre-

processed data as the input data of RoBERTa and Tf-Idf. In the second step, we get the output result of the last layer of RoBERTa (RoBERTa Output) and the output result of Tf-Idf (Tf-Idf Output). In the third step, we use the output result of Tf-Idf to weight the output result of RoBERTa. We can get a weighted result, we call it RoBERTa weighted output. In the fourth step, we connect the RoBERTa output result and the RoBERTa weighted output result together. In the fifth step, we use the result of the previous step as the input of the classifier. Use the classifier to output the prediction results of the model. In the final step, the results of the model prediction are processed into the format required by the task organizer team.

Among them, the shape of RoBERTa output [batch\_size, max\_sequence\_length, hidden\_size]. The shape of Tf-Idf output is [batch\_size, max\_sequence\_length]. Equation 1-3 is the process of weighting operation.

In equation 1,  $[Tf - Idf\_Output]_i$  is the result of the  $i - th$  batch of Tf-Idf output.  $[RoBERTa\_Output]_i$  is the result of the  $i - th$  batch of RoBERTa output. The result of multiplying these two matrices is  $[Weighted]_i$ .

In equation 2,  $[Tf - Idf\_Output]_i^T$  is the transpose of  $[Tf - Idf\_Output]_i$  matrix. The result of multiplying  $[Tf - Idf\_Output]_i^T$  and  $[Weighted]_i$  is  $[RoBERTa\_Weighted\_Output]_i$ .

In equation 3, The value range of  $i$  is an integer between 0 and batch\_size. Calculate the value of each  $[RoBERTa\_Weighted\_Output]_i$  to get  $[RoBERTa\_Weighted\_Output]$ . Its shape is the same as RoBERTa output.

Figure 3 shows the model structure and data flow

Method	F1 Score
ALBERT+Tf-Idf	82.53
BERT+Tf-Idf	82.04
RoBERTa	83.41
RoBERTa+Tf-Idf	84.81

Table 2: F1 result scores obtained on the validation set using different models. The validation set is provided by the task organizer team.

of RoBERTa combined with Tf-Idf.

### 3 Experiment and Results

In this section, we will introduce the data preprocessing methods and experimental settings we used in the task and the final results.

#### 3.1 Data Preprocessing

Combined with our analysis in the data description section, we remove the stop words of sentence pairs in the data. For the stop word list, we use the stopwords package provided by NLTK. To use the Tf-Idf algorithm to obtain the weighted output, and to ensure that the shape of the text encoding processed by the Tf-Idf algorithm is consistent with the output shape of RoBERTa, we have deleted the part of the text encoding that exceeds the maximum sentence length. For those less than the maximum sentence length for text encoding, we perform zero-padding operations. The encoding of Tf-Idf is obtained using the toolkit provided by gsim (Řehůřek and Sojka, 2010)<sup>2</sup>.

In the data input, we use the [SEP] symbol to separate the sentence pairs together. Then use the [SEP] symbol to concatenate Lemma that appears in each sentence in the sentence pair. It should be noted that the three models we used in the experiment, BERT, ALBERT, and RoBERTa, are different in the division of symbols. Here, we use [CLS] and [SEP] uniformly for the convenience of description.

#### 3.2 Experiment setting

As we introduced in the previous section, on the data set for this task, we use 4 different models to experiment with the result scores on the validation set. We adjust the parameters as much as possible to achieve the optimal results of each different model, so different models use different parameter combination settings.

<sup>2</sup><https://github.com/RaRe-Technologies/gensim>

Team	F1 Score	Rank
jaymundra	93.30	1
rohangpt	93.30	1
oyx	93.30	1
rohangpt	93.20	2
dipakam	92.80	3
LucasHub(our team 'hub')	84.60	49

Table 3: In the result list released by the task organizer team, the top 3 submitted test set prediction results scores and our submitted test set prediction results scores. There are a total of 175 results on the leaderboard of the English task. There are a total of 87 places from the first to the last.

- ALBERT+Tf-Idf: The epoch, batch size, maximum sequence length, and learning rate for the model are 6, 32, 150, and 3e-5, respectively.
- BERT+Tf-Idf: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 150, and 4e-5, respectively.
- RoBERTa+Tf-Idf: The epoch, batch size, maximum sequence length, and learning rate for the model are 5, 32, 150, and 3e-5, respectively.
- RoBERTa: The epoch, batch size, maximum sequence length, and learning rate for the model are 5, 32, 150, and 3e-5, respectively.

## 4 Results

The final result score evaluation index uses the F1 score. Therefore, the effects of the different models we used in the experimental phase are all using F1 scores to determine which model is better.

We use the same validation set data to evaluate the performance of different models. Comparing the result score obtained by the combination of ALBERT, BERT and Tf-Idf with the score obtained by the combination of RoBERTa and Tf-Idf, it can be seen that the combination strategy of RoBERTa can get a better F1 score. Compared with the F1 score obtained by using RoBERTa alone, the F1 score obtained by RoBERTa+Tf-Idf is better. This also verifies the feasibility and effectiveness of our method. We sort the results according to Table 2.

The prediction result of the English test set we finally submitted is predicted by RoBERTa+Tf-Idf.

Compared with the F1 scores obtained by the top three teams in the English data, there is still a certain gap. Our F1 score ranks middle among all result scores. Our final ranking is 49th. We sort the results according to Table 3.

## 5 Conclusion

This paper proposes a model that combines RoBERTa and Tf-Idf to calculate whether the target words in English sentence pairs are similar. We introduced our analysis of the data, the methods used in the experiment, and the results of the experiment in Sections 3 and 4. We compared the effects of different models of ALBERT, BERT, RoBERTa and the combination of Tf-Idf. The experimental results also prove that RoBERTa+Tf-Idf can get better results in our method. In future work, we will improve our methods to get better results. For example, other types of word embedding vectors can be introduced into our model, and the method of weighting and vector fusion can also be improved.

## References

- Oscar Araque, Ganggao Zhu, and Carlos A Iglesias. 2019. A semantic similarity-based perspective of affect lexicons for sentiment analysis. *Knowledge-Based Systems*, 165:346–359.
- Weilong Chen, Xin Yuan, Sai Zhang, Jiehui Wu, Yanru Zhang, and Yan Wang. 2020. *Ferryman at SemEval-2020 task 3: Bert with TFIDF-weighting for predicting the effect of context in word similarity*. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 281–285, Barcelona (online). International Committee for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Sachin Mathur and Deendayal Dinakarpandian. 2012. Finding disease similarity based on implicit semantic similarity. *Journal of biomedical informatics*, 45(2):363–371.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.
- Michael Mohler and Rada Mihalcea. 2009. Text-to-text semantic similarity for automatic short answer grading. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 567–575.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. <http://is.muni.cz/publication/884893/en>.
- Philip Resnik. 1995. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*.
- David Sánchez, Montserrat Batet, David Isern, and Aida Valls. 2012. Ontology-based semantic similarity: A new feature-based approach. *Expert systems with applications*, 39(9):7718–7728.
- Xingjian Shi, Zhouong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. 2015. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

# Lotus at SemEval-2021 Task 2: Combination of BERT and Paraphrasing for English Word Sense Disambiguation

**Niloofar Ranjbar**

Department of Computer Engineering  
Jam School of Engineering  
Persian Gulf University  
nranjbar@pgu.ac.ir

**Hossein Zeinali**

Department of Computer Engineering  
Amirkabir University of Technology  
hzeinali@aut.ac.ir

## Abstract

In this paper, we describe our proposed methods for the multilingual word-in-Context disambiguation task in SemEval-2021. In this task, systems should determine whether a word that occurs in two different sentences is used with the same meaning or not. We proposed several methods using a pre-trained BERT model. In two of them, we paraphrased sentences and add them as input to the BERT, and in one of them, we used WordNet to add some extra lexical information. We evaluated our proposed methods on test data in SemEval-2021 task 2.

## 1 Introduction

Measuring semantic similarity is a task that is important in many applications such as summarization, plagiarism detection, etc. To measure the semantic similarity between two words or sentences, the words' meaning in their contexts should be understood. In "Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation" of SemEval 2021 (Martelli et al., 2021), systems should determine whether a word that occurs in two different sentences is used with the same meaning or not. The considered languages in the Word-in-Context Disambiguation task of the evaluation are Arabic, Chinese, English, French, and Russian (Martelli et al., 2021). Due to our internal limitation, we only evaluated the proposed methods on the English dataset.

In most cases, humans can understand the correct meaning of each word by paying attention to the context of that word. This was the main reason for proposing the attention mechanism for machine translation (Vaswani et al., 2017). Because attention plays an important role in BERT topology (Devlin et al., 2019), we believe it should work for the WSD task. However, in some cases, the two sentences may be about the same subject, while

the target word has a different meaning. In these cases, find out that whether the target word has the same meaning in both sentences or not is difficult. To overcome this challenge, we need to add some extra information about the contexts to the input of the BERT.

We proposed four main methods for this task. In the first method, we use the BERT model as a language representation model in which the inputs are the two sentences that come in a row and are separated by a "[SEP]" token. Furthermore, the target word is surrounded by a "[TGT]" token in both sentences. Then, the first output of the last layer is used as a classifier to determine whether the word has the same meaning in the two sentences or not. In the second one, we added some information about the target word from the WordNet<sup>1</sup> to the end of each input and used the same strategy as the first method. In the third and fourth methods, we used the paraphrases of sentences as additional inputs to the BERT.

The remainder of the paper is organized as follows: In section 2, we provide a short literature review of the WSD task and mention some related works that used BERT for this task. After that, in section 3, we introduce our methods in consequent sections: the first method that only uses the BERT model is explained in Section 3.1. The second method that uses the BERT model and some additional information from WordNet is explained in Section 3.2. The third and fourth methods that use the BERT model and add paraphrased sentences as an additional input to the model are explained in Section 3.3. Then in Section 4, the used datasets in our experiments are specified. Finally, the results and discussion are presented in Section 5.

---

<sup>1</sup><https://wordnet.princeton.edu/>



## 2 Background

### 2.1 Task Setup

The dataset provided by MCL-WiC<sup>2</sup> addresses both multilingual and cross-lingual aspects, covers all part-of-speeches, and also a high number of domains and genres. The task of the challenge is a binary classification where systems should specify whether the target word has the same meaning in both sentences (T for true) or different meaning (F for false). The sentences can be from the same language (multilingual dataset) or across different languages (cross-lingual dataset) (Martelli et al., 2021).

In the following you can find an example of sentence pair in English from the multilingual dataset<sup>3</sup>:

- In that context of coordination and integration, Bolivia holds a key play in any process of infrastructure development.
- A musical play on the same subject was also staged in Kathmandu for three days.

In this example, the target word “play” should be tagged as F (False) since it is used in two distinct meanings.

We evaluated our methods on the English part of the multilingual dataset. We used the MCL-WiC training set, containing 8000 sentence pairs in English as the training data, and the MCL-WiC development set containing 1000 sentence pairs in English as the validation set for our system.

### 2.2 Related works

There are several published works that used BERT for WSD task (Wiedemann et al., 2019; Du et al., 2019; Yap et al., 2020; Guo et al., 2020) and also some other papers that combined BERT and WordNet for this task (Levine et al., 2020; Peters et al., 2019).

For example, (Levine et al., 2020) proposed a method named SenseBERT in which the model is pre-trained to predict the masked words and their WordNet super-senses. They proposed a mechanism in which they pay attention to the words at the sense level. Therefore, they achieved a lexical-semantic level language model. SenseBERT achieved 72.1 % accuracy on the “Word in

Context” task which is a state-of-the-art result on this task.

(Peters et al., 2019) proposed a general method in which multiple knowledge bases are embedded into large-scale models. By doing that and using structured knowledge, they enhanced their representations. First, they use an entity linker to retrieve relevant entity embeddings for each knowledge base (KB). After that, they use word-to-entity attention to update contextual word representations. They trained entity linkers and self-supervised language modeling together in an end-to-end multi-task setting in which a large amount of raw text is combined with a small amount of entity linking supervision. By merging WordNet and a subset of Wikipedia into BERT, KnowBert shows improvement in several tasks such as word sense disambiguation. This method achieved a good result on the “Word in Context” task with 70.9 % accuracy.

“GlossBERT” is another work that was proposed in (Huang et al., 2019). The contexts and glosses of the target words were put together and considered as inputs to the BERT. Three BERT-based models were proposed for WSD. The SemCor3.0 training corpus was used to fine-tune the pre-trained BERT model and finally, the models were evaluated on several English WSD benchmark datasets. The experimental results show that the proposed method achieved new state-of-the-art results on the WSD task.

## 3 System Overview

In this section, we explain our proposed methods. The first method that relies only on the BERT model is explained in Section 3.1. This method is further improved in 3.2 by adding some extra information extracted from the WordNet to the BERT’s input. Finally, in section 3.3 we explain an augmentation method by generating the paraphrases of both sentences and add them as an extra input to the BERT.

### 3.1 BERT Method

In this method, we fine-tune a BERT model using the TensorFlow-models PIP package as explained in [fine-tuning-BERT Tutorial](#). To do this, we use the pre-trained BERT encoder (large-uncased-BERT) from [TensorFlow Hub](#). As mentioned in the tutorial, to fine-tune a pre-trained model, exactly the same tokenization, vocabulary, and index mapping with the model should be used. So we

<sup>2</sup>Multilingual and Cross-lingual Word-in-Context Disambiguation task

<sup>3</sup>[https://github.com/SapienzaNLP/mcl-wic/blob/master/SemEval-2021\\_MCL-WiC\\_all-datasets.zip](https://github.com/SapienzaNLP/mcl-wic/blob/master/SemEval-2021_MCL-WiC_all-datasets.zip)

rebuild and use the tokenizer that was used by the base model.

Next, we preprocess the provided data and prepare it as the input of the BERT model. For every pair of sentences first, we add a “[TGT]” token before and after the target word in both sentences. After that, we add a “[SEP]” (Separator) token at the end of both sentences. Then we encode them separately by the tokenizer. The tokenizer itself changes all words to lowercase and separates the unknown words. We also encode and add a “[CLS]” token at the first position of each example to be able to do a classification task. Examples are constructed by concatenation of two sentences. Note that we had to add “[TGT]” as a token into the tokenizer vocabulary. Here is an example of a pair of sentences:

```
[CLS] In that context of coordination and
integration, Bolivia holds a key [TGT]
play [TGT] in any process of infras-
tructure development. [SEP] A musical
[TGT] play [TGT] on the same subject
was also staged in Kathmandu for three
days.[SEP]
```

The output of the tokenization process is named input word ids. For the above example, we will have:

```
[ 101 1999 2008 6123 1997 12016 1998
8346 1010 11645 4324 1037 3145 1
2377 1 1999 2151 2832 1997 6502 2458
1012 102 1037 3315 1 2377 1 2006 1996
2168 3395 2001 2036 9813 1999 28045
2005 2093 2420 1012 102 ]
```

Note that input word ids vectors should be padded with zero token ids to make them equal length.

We also add two vectors as extra inputs to the BERT model. One of them is called input mask, in which, for every non-padding token, we put “1”, and for every padding token, we put “0”. Another one is called input type ids in which, for each token of the first sentence, second sentence, and padding tokens, we put “1”, “2”, and “0” respectively. Thus, the input vectors equivalent to the above sentence pairs (without padding tokens) are as follows:

```
input_word_ids: [101 1999 2008 6123
1997 12016 1998 8346 1010 11645 4324
1037 3145 1 2377 1 1999 2151 2832
```

```
1997 6502 2458 1012 102 1037 3315 1
2377 1 2006 1996 2168 3395 2001 2036
9813 1999 28045 2005 2093 2420 1012
102 ]
```

```
input_mask: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 ]
```

```
input_type_ids: [0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 ]
```

After preparing the input, we define the BERT classifier with the following configuration:

```
'attention_probs_dropout_prob': 0.1,
'hidden_act': 'gelu',
'hidden_dropout_prob': 0.1,
'hidden_size': 768,
'initializer_range': 0.02,
'intermediate_size': 3072,
'max_position_embeddings': 512,
'num_attention_heads': 12,
'num_hidden_layers': 12,
'type_vocab_size': 2,
'vocab_size': 30522
```

As mentioned in [fine-tuning-BERT Tutorial](#) : “BERT adopts the Adam optimizer with weight decay. It uses a learning rate that firstly warms up from 0 and then decreases to 0.” We choose the batch size= 32 and train the BERT classifier with maximum epochs=10. We also use an early stopping call back, which is based on the validation set loss, and make the training process stopped, when the validation loss increase in two steps.

### 3.2 BERT plus WordNet method

As mentioned before, in some WSD cases using just the target word context can not be helpful, so we proposed a new method in which some information from the WordNet is added to the end of each sentence.

In this method, for every synset of the target word, we define a score that indicates how much that synset is related to the target word in this context. To calculate this score, we used the following equation:

$$path\_dist = \sum_{w \in context} \frac{1}{position\_dist(w, target\_w) \times \min_{syn_2 \in syns(w)} (spd(syn_1, syn_2))} \quad (1)$$

where *position\_dist* indicates how much the target word is far from the word “*w*” in the context. So, the farther the word is, the effect of that to calculate the path dist is less. “*syns(w)*” are all synsets of the word “*w*” in the context. “*syn<sub>1</sub>*” is the synset of the target word that we want to calculate its path distance. Note that, before calculating the score, we remove all stop words, punctuations, and numbers from the context of the target word. *spd* is a function that is already implemented in NLTK package<sup>4</sup>. It returns the distance of the shortest path linking the two synsets (if one exists). For each synset, all the ancestor nodes and their distances are recorded and compared. The ancestor node common to both synsets that can be reached with the minimum number of traversals is used. If no ancestor nodes are common, -1 is returned. If a node is compared with itself, 0 is returned<sup>5</sup>.

As can be seen, if the *path\_dist* increases, the synset is less related to the target word in this context. Therefore, we sort the synsets based on how relevant they are to the target word and use their Lexname<sup>6</sup> (which is called supersense in (Levine et al., 2020)) to represent them. After that, we add all the sorted Lexnames at the end of each sentence separated by a “[LX]” token.

Here is an example for a pair of sentences:

**sentence1:** in that context of coordination and integration , bolivia holds a key [TGT] play [TGT] in any process of infrastructure development .[LX] time [LX] state [LX] act [LX] communication [LX] event [LX] attribute [SEP]

**sentence2:** a musical [TGT] play [TGT] on the same subject was also staged in kathmandu for three days .[LX] time [LX] communication [LX] act [LX] state [LX] attribute [SEP]

Finally, we define *input\_word\_ids*, *input\_mask*, and *input\_type\_ids* same as in Section 3.1 and train the model defined in that section.

<sup>4</sup>[https://www.nltk.org/\\_modules/nltk/corpus/reader/wordnet.html](https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html)

<sup>5</sup><https://docs.huihoo.com/nltk/0.9.5/api/nltk.wordnet.synset.Synset-class.html>

<sup>6</sup>More information about Lexname can be found at <https://wordnet.princeton.edu/documentation/lexnames5wn>

### 3.3 BERT plus Paraphrase Method

As paraphrasing changes the structure and some words of a sentence while the meaning remains the same, it can be helpful to add some extra information as input to the BERT encoder. By adding this information, we help the model to see other meanings of the context words.

We used Machine Translation to generate paraphrases of the sentences. To do that, we use Microsoft Translator Text API<sup>7</sup> to translate all of the sentences from English to Spanish and then translate them back from Spanish to English. As this API has reliable results on translating from and to the Spanish language, we choose it as the intermediate language. However, other languages such as French and Germany can be used as well.

Based on this idea, two methods are proposed in this section:

#### 3.3.1 Using generated paraphrases

In the first method, we add the paraphrased sentences word ids, mask, and type\_ids as additional inputs to the BERT encoder. Thus, the input to the BERT encoder is constructed from 6 parts instead of 3 parts defined in Section 3.1. Normally, three input vectors are used for the BERT. In this case, we add three additional input vectors which are made with the paraphrase of sentences instead of sentences themselves. In our opinion, the best way of combining this new data is to concatenate paraphrases to the main sentences, but in this work, due to our internal limitations, we could not do this, and instead, we extended the three inputs to six. In practice, the input is the sum of the embedded vectors of the six vectors. We do not use the “[TGT]” token before and after the target word in the generated paraphrase because it may be changed.

#### 3.3.2 Using just the different words

As the generated paraphrases are not very different from the original sentences most of the time, we propose another method in which we just find the words that are changed in the generated paraphrase and then concatenate them with the original words in the original sentence with a “[S]” token. To do that, we should find the original word of each newly generated word by finding the most similar word from the main sentence to the generated word. We used the *fastText*<sup>8</sup> pre-trained model in English to

<sup>7</sup><https://www.microsoft.com/en-us/translator/business/translator-api/>

<sup>8</sup><https://fasttext.cc>

Methods	run1	run2	run3	run4	run5	Average	Std
BERT Method (3.1)	84.4	84.9	84.2	85.8	83.8	<b>84.62</b>	0.688
BERT + WordNet (3.2)	84.5	84.7	84.3	85.2	85.2	<b>84.78</b>	0.366
BERT + Paraphrase Sents (3.3.1)	85.3	85.6	85.6	86.9	85.5	<b>85.78</b>	0.571
BERT + Paraphrase Words (3.3.2)	84.1	86.3	85.6	85.4	82.4	<b>84.76</b>	1.378

Table 1: Results of the proposed methods in 5 different runs. Std in the last column indicate *Standard Deviation*.

find the most similar word, and then consider it as the paraphrased word.

## 4 Experimental Setup

As mentioned before, we used the MCL-WiC training set, containing 8000 pair sentences in English as the training data, and the MCL-WiC development set containing 1000 pair sentences in English as the validation data for our system. Similarly, the MCL-WiC test set containing 1000 pair sentences in English was used as the test data. The results displayed in Table 1 are based on the test set. We use the validation set only for finding the best number of fine-tuning iterations. It is worth mentioning that in test phase of the third method, we first paraphrase the sentences of the test set and make them ready for the model input as we did for the training and validation sets. All reported results are based on accuracy that is the main criteria for the challenge.

## 5 Results

Due to the random initialization of classifier layer weights, different results were achieved from different runs. So, we decided to run each experiment 5 times and report the average and standard deviation of model accuracy.

Table 1 shows the obtained results from different methods based on the accuracy percentage. It is obvious that adding generated sentence level paraphrases proposed in Section 3.3.1 improves the average accuracy compared to the BERT method. However, we got only a marginal improvement by adding paraphrased words proposed in Section 3.3.2. We guess the reason for this has two folds: first, we had some difficulties in finding the original word of the changed word, and using only the fastText may not be helpful. For example, the word “play” changed to “work” in the generated paraphrase, while using the fastText, the system found another word instead of “work”. The second reason can be eliminating not-changed words that cause a

discontinuity in the word context.

On the other hand, adding more information from WordNet proposed in Section 3.2 is not very helpful, and the accuracy is not changed compared to the base method proposed in Section 3.1. This has happened because of the used way for adding this information to the input and the shape of information. For example, adding only the two first synsets or adding lemmas or glosses of synsets instead of their lexnames may cause better results.

It is clear that there is a variation in the results of each method for different runs. The last column of the table shows the standard deviation of the five obtained results for each method. The second and third methods have smaller STD that means these methods are robust to random initialization classifier’s weights, and we can trust more to the results.

In summary, adding generated paraphrases from sentences increase the performance of the model. However, using a better method for generating paraphrases can lead to better results.

## 5.1 Conclusion

In this work, we proposed four BERT-based methods for the WSD task. To handle some issues such as the same contexts for different word meanings, we proposed to add some extra information from WordNet or generate paraphrases of sentences. We trained and evaluated our proposed methods on the MCL-WiC training and testing sets, respectively. The results show that adding generated paraphrases as an additional input to the BERT can be helpful. Although, the performance of the paraphrase generation method plays an important role. So, for future works, using different methods of generating paraphrases can be considered. Besides, as mentioned before, adding only the two first synsets or adding lemmas or glosses of synsets instead of their lexnames from WordNet can be investigated. Due to lack of time, we could not spend too much effort on this challenge, and we leave it for the future.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using BERT for word sense disambiguation. *arXiv preprint arXiv:1909.08358*.
- Ping Guo, Yue Hu, and Yunpeng Li. 2020. MG-BERT: A multi-glosses BERT model for word sense disambiguation. In *International Conference on Knowledge Science, Engineering and Management*, pages 263–275. Springer.
- Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. [GlossBERT: BERT for word sense disambiguation with gloss knowledge](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.
- Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. [SenseBERT: Driving some sense into BERT](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. [Knowledge enhanced contextual word representations](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- Gregor Wiedemann, Steffen Remus, Avi Chawla, and Chris Biemann. 2019. Does BERT make any sense? interpretable word sense disambiguation with contextualized embeddings. *arXiv preprint arXiv:1909.10430*.
- Boon Peng Yap, Andrew Koh, and Eng Siong Chng. 2020. [Adapting BERT for word sense disambiguation with gloss selection objective and example sentences](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 41–46, Online. Association for Computational Linguistics.

# Cambridge at SemEval-2021 Task 2: Neural WiC-Model with Data Augmentation and Exploration of Representation

Zheng Yuan and David Strohmaier

ALTA Institute, University of Cambridge, United Kingdom

Department of Computer Science and Technology, University of Cambridge, United Kingdom

{zheng.yuan, david.strohmaier}@cl.cam.ac.uk

## Abstract

This paper describes the system of the Cambridge team submitted to the SemEval-2021 shared task on Multilingual and Cross-lingual Word-in-Context Disambiguation. Building on top of a pre-trained masked language model, our system is first pre-trained on out-of-domain data, and then fine-tuned on in-domain data. We demonstrate the effectiveness of the proposed two-step training strategy and the benefits of data augmentation from both existing examples and new resources. We further investigate different representations and show that the addition of distance-based features is helpful in the word-in-context disambiguation task. Our system yields highly competitive results in the cross-lingual track without training on any cross-lingual data; and achieves state-of-the-art results in the multilingual track, ranking first in two languages (Arabic and Russian) and second in French out of 171 submitted systems.

## 1 Introduction

Polysemy still poses a great challenge to natural language processing (NLP) applications. Depending on its context, an ambiguous word can refer to multiple, potentially unrelated, meanings. Recently, as an application of Word Sense Disambiguation (WSD) (Navigli, 2009, 2012), Word-in-Context (WiC) disambiguation has been framed as a binary classification task to identify if the occurrences of a target word in two contexts correspond to the same meaning or not. The release of the WiC dataset (Pilehvar and Camacho-Collados, 2019), followed by the Multilingual Word-in-Context (XL-WiC) dataset (Raganato et al., 2020), has helped provide a common ground for evaluating and comparing systems while encouraging research in WSD and context-sensitive word embeddings.

In this paper, we describe our submission to the SemEval-2021 shared task on Multilingual and Cross-lingual Word-in-Context (MCL-WiC) Disambiguation (Martelli et al., 2021), which involves determining whether a word shared by two sentences in the same language (multilingual track) or across different languages (cross-lingual track) has the same meaning in both contexts. Compared to previous WiC and XL-WiC benchmarks, two new languages are introduced as well as a cross-lingual track where systems are evaluated under a ‘zero-shot’ setting.

The MCL-WiC task directly classifies pairs of sentences with regard to the meaning of the shared word. By turning WSD into a binary comparison task, MCL-WiC avoids the need for sense tags of previous WSD shared tasks (Manandhar et al., 2010; Navigli et al., 2013; Moro and Navigli, 2015). It also resembles the Word Sense Alignment (WSA) task (Ahmadi et al., 2020) more closely, in which definitions from different dictionaries have to be aligned. Contextualised word embeddings and pre-trained Transformer-based (Vaswani et al., 2017) language models have been increasingly applied to these tasks and state-of-the-art results have been reported (Hadiwinoto et al., 2019; Vial et al., 2019; Levine et al., 2020; Raganato et al., 2020; Pais et al., 2020; Manna et al., 2020; Lenka and Seung-Bin, 2020).

In line with previous research, we develop a neural system based on pre-trained multilingual masked language model XLM-R (Conneau et al., 2020). Additionally, we introduce three distance-based features to be used together with the widely used sequence and token representations for MCL-WiC disambiguation. To further improve system performance, we apply automatic data augmentation and extract examples from multiple external resources. A two-step training strategy is then employed to make use of both in-domain and out-of-

Split	Multilingual					Cross-lingual			
	EN-EN	AR-AR	FR-FR	RU-RU	ZH-ZH	EN-AR	EN-FR	EN-RU	EN-ZH
Train	8,000	-	-	-	-	-	-	-	-
Dev	1,000	1,000	1,000	1,000	1,000	-	-	-	-
Test	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000

Table 1: Number of instances in the official MCL-WiC dataset in each multilingual and cross-lingual sub-track.

Example	Instance	Sentence #1	Sentence #2	Label
(a)	Existing	There is never any point in trying to make oneself heard over <b>noise</b> .	We have formulated a programme to address the traffic <b>noise</b> impact of existing roads.	T
	Existing	There is never any point in trying to make oneself heard over <b>noise</b> .	He went to bed but could not fall asleep because of the <b>noise</b> .	T
	New	We have formulated a programme to address the traffic <b>noise</b> impact of existing roads.	He went to bed but could not fall asleep because of the <b>noise</b> .	T
(b)	Existing	Wages have declined sharply as a proportion of the <b>subsistence</b> minimum.	Agriculture, largely of a <b>subsistence</b> nature, is the main economic activity.	T
	Existing	Wages have declined sharply as a proportion of the <b>subsistence</b> minimum.	The third member of the Committee is paid a daily fee for each working day plus <b>subsistence</b> allowance.	F
	New	Agriculture, largely of a <b>subsistence</b> nature, is the main economic activity.	The third member of the Committee is paid a daily fee for each working day plus <b>subsistence</b> allowance.	F

Table 2: Sample sentence-pair instances selected for data augmentation. The target words are marked in bold. Examples are extracted from the MCL-WiC dataset.

domain<sup>1</sup> data.

In the remainder of the paper, we present the MCL-WiC disambiguation shared task in Section 2 and our approach in Section 3. In Section 4, we describe the experiments and present results on the development set. Section 5 summarises the official evaluation results. Finally, we provide an analysis of our system in Section 6 and conclude in Section 7.

## 2 Task Description

The MCL-WiC dataset used in the shared task consists of sentence pairs sharing the same target word in the same language or across different languages. The task considers five languages: Arabic (AR), Chinese (ZH), English (EN), French (FR) and Russian (RU); and contains five multilingual (EN-EN, AR-AR, FR-FR, RU-RU, ZH-ZH)<sup>2</sup> and four cross-lingual (EN-AR, EN-FR, EN-RU, EN-ZH) sub-tracks. Training data is available for the multilingual EN-EN sub-track only, and development data is available for all five multilingual sub-tracks. No cross-lingual training or development data is provided. Statistics of the MCL-WiC dataset are

<sup>1</sup>In this paper, we use the term ‘out-of-domain’ to refer to data from additional resources (i.e. not provided by the shared task organisers) - see Section 4.1 for more details.

<sup>2</sup>[language of sentence #1]-[language of sentence #2]

presented in Table 1.

Results are computed using the accuracy measure, i.e. the ratio of correctly predicted instances (true positives or true negatives) to the total number of instances.

## 3 Approach

### 3.1 Data augmentation

Each instance in the (\*)WiC datasets (i.e. WiC, XL-WiC and MCL-WiC) is composed of a target word and two sentences in which the target word occurs. We notice that there are cases where the same sentence appears in multiple instances. As shown in Table 2, two existing instances, which share the same target word, contain the same first sentence, but different second sentences. Therefore, we construct new instances by pairing the second sentences from these existing instances and assign labels based on the original labels:

- If both existing instances are positive (‘T’, i.e. the target word is used in the same meaning), the resulting instance should be positive (‘T’) as well - see Example (a) in Table 2:

$$M(w_{s_1}) = M(w_{s_2}), M(w_{s_1}) = M(w_{s_3}) \\ \Rightarrow M(w_{s_2}) = M(w_{s_3})$$

where  $M(w_{s_n})$  indicates the meaning of the target word ( $w$ ) used in sentence  $n$  ( $s_n$ ).

- If one of them is positive (‘T’) and the other is negative (‘F’, i.e. the target word is used in a different meaning), the new instance should then be negative (‘F’) - see Example (b) in Table 2:

$$M(w_{s_1}) = M(w_{s_2}), M(w_{s_1}) \neq M(w_{s_3}) \\ \Rightarrow M(w_{s_2}) \neq M(w_{s_3})$$

### 3.2 Model

Following Raganato et al. (2020), we use pre-trained XLM-R as the underlying language model, which is a Transformer-based multilingual masked language model that has been trained on one hundred languages (Conneau et al., 2020). Unlike previous WiC and XL-WiC models employing a logistic regression classifier (Wang et al., 2019; Raganato et al., 2020), we add two additional layers on top of the Transformer-based model to perform classification: a linear layer with *tanh* activation, followed by another linear layer with *sigmoid* activation.

The model takes as input the two sentences in each instance. For the representation to be fed into the linear layers, we concatenate the representation corresponding to the first special token ( $[s]$ ) of the input sequence,<sup>3</sup> the vector representations of the target word in the first ( $[w_{s_1}]$ ) and second sentences ( $[w_{s_2}]$ ), as well as the element-wise absolute difference, cosine similarity (*cos*) and Euclidean distance (*dist*) between these two vectors:

$$[s; w_{s_1}; w_{s_2}; |w_{s_1} - w_{s_2}|; \cos; \text{dist}] \quad (1)$$

For those cases where the target word is split into multiple sub-tokens, we take the averaged representation rather than the representation of its first sub-token, which has been used in previous work (Wang et al., 2019; Raganato et al., 2020).<sup>4</sup>

### 3.3 Training strategy

Inspired by the success of multi-stage training for tasks like grammatical error correction (Kiyono et al., 2019; Omelianchuk et al., 2020; Yuan and Bryant, 2021) and machine translation (Zhang et al., 2018), we employ a two-step training strategy: 1) pre-training on out-of-domain data; and 2) fine-tuning with in-domain MCL-WiC data.

<sup>3</sup>The  $[s]$  token in XLM-R is equivalent to the  $[CLS]$  token in BERT (Devlin et al., 2019).

<sup>4</sup>Our preliminary experiments show that using the averaged representation is more effective than that of the first sub-token.

<b>Sentence #1</b>	I went outside to get some fresh <b>air</b> .	(A2)
<b>Sentence #2</b>	He has an <b>air</b> of authority.	(C2)
<b>Label</b>	F	

Table 3: A sentence-pair example extracted from CALD. The target words are marked in bold. A2: elementary English, C2: proficiency English.

## 4 Experiments

### 4.1 Data

In addition to the MCL-WiC dataset provided by the shared task organisers, we introduce two types of out-of-domain data: 1) (\*)WiC datasets: WiC and XL-WiC; and 2) sentence pairs constructed with examples extracted from datasets that have been annotated with both complexity and sense information: the Cambridge Advanced Learner’s Dictionary (CALD)<sup>5</sup> and SeCoDa (Strohmaier et al., 2020).

**WiC** The English WiC dataset was created using example sentences from WordNet (Fellbaum, 1998), VerbNet (Kipper-Schuler, 2005), and Wiktionary. We extract 6,066 labelled instances (by removing those without gold labels) and use them for the shared task.

**XL-WiC** The XL-WiC dataset extends the WiC dataset to 12 more languages from two resources: multilingual WordNet for ZH, Bulgarian (BG), Croatian (HR), Danish (DA), Dutch (NL), Estonian (ET), Farsi (FA), Japanese (JA) and Korean (KO); and multilingual Wiktionary for FR, German (DE), Italian (IT). In total, the XL-WiC dataset contains 112,430 labelled non-English sentence pairs, including 3,046 ZH-ZH instances and 48,106 FR-FR ones.<sup>6</sup> In contrast to the MCL-WiC task, the XL-WiC dataset does not include any cross-lingual sentence pairs.

**CALD** The CALD contains information about which words and which meanings of those words are known and used by learners at each Common European Framework of Reference (CEFR) level from A1 (beginner) to C2 (proficiency English). Only example sentences sharing the same target word, that is used in a different meaning as well as at a different CEFR level, are paired. In this

<sup>5</sup><https://www.englishprofile.org/wordlists/evp>

<sup>6</sup>In the XL-WiC dataset, the number of instances varies considerably by language - see Raganato et al. (2020) for more details.



Dataset	Language	#instances (w/o aug.)	#instances (with aug.)	Training stage
WiC	EN-EN	6,066	8,276	pre-training
XL-WiC(FR)	FR-FR	48,016	54,771	pre-training
XL-WiC(ZH)	ZH-ZH	3,046	3,370	pre-training
XL-WiC	All*	112,430	127,765	pre-training
CALD	EN-EN	34,205	-	pre-training
SeCoDa	EN-EN	10,712	-	pre-training
MCL-WiC-train	EN-EN	8,000	10,798	fine-tuning
MCL-WiC-dev(EN)	EN-EN	1,000	-	development

Table 4: Summary of datasets used in our experiments. All\*: FR-FR, ZH-ZH, BG-BG, HR-HR, DA-DA, NL-NL, ET-ET, FA-FA, JA-JA, KO-KO, DE-DE, and IT-IT; w/o aug.: without data augmentation; with aug.: with data augmentation.

#	Pre-training data	EN-EN	AR-AR	FR-FR	RU-RU	ZH-ZH	Avg.
1	WiC <sub>aug.</sub> + XL-WiC(FR&ZH) <sub>aug.</sub>	89.90	78.10	80.10	83.60	78.20	81.98
2	WiC <sub>aug.</sub> + XL-WiC <sub>aug.</sub>	89.60	79.20	82.90	85.40	<b>79.30</b>	83.28
3	WiC <sub>aug.</sub> + XL-WiC <sub>aug.</sub> + CALD + SeCoDa	90.30	<b>80.20</b>	<b>83.30</b>	<b>86.30</b>	76.90	83.40
4	Ensemble (MV)	<b>90.80</b>	79.70	83.00	85.40	<b>79.30</b>	<b>83.64</b>

Table 5: Performance of individual systems and the ensemble on MCL-WiC-dev (multilingual track). The best results in each sub-track are marked in bold. Avg.: averaged accuracy.

way, sentence pairs are encoded with additional word complexity information. In total, we generate 34,205 negative EN-EN instances.<sup>7</sup> An example is given in Table 3.

**SeCoDa** is an English language corpus of words annotated with both complexity and word senses. The original data comes from three sources: professional News, WikiNews and Wikipedia articles. The senses are drawn from the CALD and come at two levels of granularity. To use this dataset for the MCL-WiC task, sentences sharing an annotated word are paired: if the word shares a sense, the pair of sentences is labelled as ‘T’; otherwise, it is labelled as ‘F’. We use the finer level of granularity for this assignment. Overall, we extract 10,712 labelled pairs (9,015 positive and 1,697 negative).

All the data introduced in this section is regarded as out-of-domain data and therefore used in the pre-training stage, and the in-domain MCL-WiC training data is used in the fine-tuning stage. For development, we use only the EN-EN MCL-WiC development set (**MCL-WiC-dev(EN)**) - see Table 4). A single model is developed to target all multilingual and cross-lingual tracks. It is worth noting that neither the multilingual AR-AR and RU-RU data nor the cross-lingual data is used for

<sup>7</sup>Due to time limitations, we have not used any positive instances from CALD and leave it for future work.

training, i.e. zero-shot.

The data augmentation method proposed in Section 3.1 is applied to the (\*)WiC datasets, but not to CALD or SeCoDa. Detailed statistics of the corpora used in our experiments are presented in Table 4.

## 4.2 Training details

In our experiments, models are trained by minimising the binary cross-entropy loss between their prediction and the gold label. We use the AdamW optimiser (Loshchilov and Hutter, 2019) with a fixed learning rate of 1e-5 for all models.<sup>8</sup> We use a dropout layer with a dropout probability of 0.2. The input texts are processed in batches of 8 and are padded or truncated to a length of 182.<sup>9</sup> We select the model with the best validation accuracy on **MCL-WiC-dev(EN)**. Each model is trained on one NVIDIA Tesla P100 GPU.

## 4.3 Results

We construct three pre-training sets using different combinations of the out-of-domain data and

<sup>8</sup>We use the pre-trained xlm-roberta-large model provided by Hugging Face (<https://huggingface.co/>) transformers library (Wolf et al., 2020).

<sup>9</sup>We use a maximum sequence length of 182, which is the length of the longest input sequence in the MCL-WiC training set.

System	EN-EN		AR-AR		FR-FR		RU-RU		ZH-ZH	
	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank
#4	92.50	6	84.80	1	86.50	2	87.40	1	85.80	13

Table 6: Official results of our best submitted system on the MCL-WiC test set (multilingual track). Acc.: accuracy.

System	EN-AR		EN-FR		EN-RU		EN-ZH	
	Acc.	Rank	Acc.	Rank	Acc.	Rank	Acc.	Rank
#1	86.50	6	85.70	10	86.80	9	88.80	5

Table 7: Official results of our best submitted system on the MCL-WiC test set (cross-lingual track).

train three systems. All these systems are fine-tuned with the augmented version of the MCL-WiC training set (**MCL-WiC-train<sub>aug.</sub>** - see Table 4). Results on the MCL-WiC multilingual development set are presented in Table 5, where **WiC<sub>aug.</sub>** is the augmented version of the WiC dataset, **XL-WiC<sub>aug.</sub>** is the augmented version of the full XL-WiC dataset for all 12 languages, and **XL-WiC(FR&ZH)<sub>aug.</sub>** is a subset of **XL-WiC<sub>aug.</sub>**, containing only examples in FR and ZH (i.e. the only two languages shared by MCL-WiC and XL-WiC).

We can see that adding pre-training examples in other languages from the XL-WiC dataset improves the results for all languages except for EN-EN, where the accuracy slightly drops from 89.90 to 89.60 (Table 5 #2). Interestingly, Raganato et al. (2020) also reported performance gains in all languages after adding multilingual data. Examples from different languages can still help models better generalise across languages. The addition of English data from CALD and SeCoDa is also beneficial, yielding further improvements in all languages except for ZH-ZH (#3). Finally, the predictions from all three systems are used in an ensemble model. For each final prediction, the majority vote (MV) of these predictions is taken, i.e. the prediction with the most votes is chosen as final prediction. The ensemble model yields the highest averaged score, as well as in EN-EN and ZH-ZH sub-tracks (#4), suggesting that all three systems (#1, #2 and #3) are complementary.

## 5 Official evaluation results

We submit our systems to all multilingual and cross-lingual tracks. The official results of our best submission for each track are reported in Table 6 and Table 7. Our ensemble system (**System #4**) achieves state of the art in the multilingual track,

ranking first in both AR-AR and RU-RU sub-tracks without any AR or RU training data, and second in FR-FR out of 171 submitted systems. For the cross-lingual track, our ‘zero-shot’ system (**System #1**) is consistently within the top ten ranks out of 171 total submissions.<sup>10</sup>

## 6 Analysis

### 6.1 Effect of two-step training

We propose a two-step training strategy to make use of both in-domain and out-of-domain data. To investigate the effectiveness of both training steps, we undertake an ablation study, in which we remove one training step at a time. Table 8 presents the ablation test results of the system pre-trained on **WiC<sub>aug.</sub> + XL-WiC(FR&ZH)<sub>aug.</sub>**, and fine-tuned on **MCL-WiC-train<sub>aug.</sub>** (i.e. **System #1** in Table 5).

The results of the ablation study demonstrate the effectiveness of the two-step training strategy, and show that it is crucial to have both pre-training and fine-tuning stages. Performance drops in all multilingual sub-tracks when removing either step, except for removing the pre-training step in FR-FR (+0.70). This is interesting as the model is pre-trained on data for only three sub-tracks, where FR-FR is one of them. For the other two languages, we observe the biggest performance decrease: ZH-ZH (-2.80) and EN-EN (-2.30). Overall, the fine-tuning stage seems more effective than the pre-training stage, though more data is used in the latter (10,798 for fine-tuning vs. 66,417 for pre-training), demonstrating the importance of having high-quality in-domain data.

<sup>10</sup>It is to be noted that the calculation of the rank counts every submission, even if they were made by the same team.

Ablated stage	EN-EN		AR-AR		FR-FR		RU-RU		ZH-ZH		Avg.	
	Acc.	$\Delta$	Acc.	$\Delta$	Acc.	$\Delta$	Acc.	$\Delta$	Acc.	$\Delta$	Acc.	$\Delta$
System #1	89.90	-	78.10	-	80.10	-	83.60	-	78.20	-	81.98	-
no pre-training	87.60	-2.30	77.10	-1.00	80.80	+0.70	82.70	-0.90	75.40	-2.80	80.72	-1.26
no fine-tuning	87.10	-2.80	75.80	-2.30	74.80	-5.30	79.60	-4.00	75.90	-2.30	78.64	-3.34

Table 8: Ablation test results of **System #1** on MCL-WiC-dev (multilingual track).  $\Delta$  denotes the difference in accuracy (Acc.) score with respect to **System #1**.

Representation	EN-EN	AR-AR	FR-FR	RU-RU	ZH-ZH	Avg.
$[s; w_{s_1}; w_{s_2}]$	84.70	<b>77.30</b>	80.10	<b>83.80</b>	67.30	78.64
$[s; w_{s_1}; w_{s_2};  w_{s_1} - w_{s_2} ]$	85.40	77.00	80.60	81.50	70.30	78.96
$[s; w_{s_1}; w_{s_2};  w_{s_1} - w_{s_2} ; \cos; \text{dist}]$	<b>87.60</b>	77.10	<b>80.80</b>	82.70	<b>75.40</b>	<b>80.72</b>

Table 9: Results of models using different representations. Systems are trained on MCL-WiC-train<sub>aug</sub> and evaluated on MCL-WiC-dev (multilingual track). The best results in each sub-track are marked in bold. *cos*: cosine similarity, *dist*: Euclidean distance.

## 6.2 Comparison of representations

In our system, the representation pooled out from the underlying pre-trained language model is a combination of three vector representations (of the first token and target word in both sentences), and three distance-based features: the element-wise absolute difference, cosine similarity and Euclidean distance between the target word in both sentences (see Section 3.2). We further experiment with different representations and present our results in Table 9. We can see that our proposed representation yields the overall best performance across different languages, suggesting that the addition of all three distance-based features is indeed helpful.

## 7 Conclusion

In this paper, we presented the contribution of the Cambridge University team to the SemEval 2021 shared task on MCL-WiC Disambiguation. Using XLM-R, a pre-trained multilingual Transformer-based language model, as a starting point, we investigated automatic data augmentation, the use of multiple external datasets, multi-stage training strategies, and the representation of tokens and their distance. Our detailed analysis demonstrated the effectiveness of the two-step training strategy for making use of both in-domain and out-of-domain data, as well as the benefits of adding distance-based features to the representation for WiC disambiguation. Our best system yields highly competitive results in the cross-lingual track and achieves state-of-the-art results in the multilingual track, ranking first in two languages (AR and RU) and second in FR out of 171 total submissions.

## Acknowledgments

We would like to thank Mariano Felice and Shiva Taslimipoor for their valuable comments and suggestions. This paper reports on research supported by Cambridge Assessment, University of Cambridge. This work was performed using resources provided by the Cambridge Service for Data Driven Discovery operated by the University of Cambridge Research Computing Service, provided by Dell EMC and Intel using Tier-2 funding from the Engineering and Physical Sciences Research Council (capital grant EP/P020259/1), and DiRAC funding from the Science and Technology Facilities Council. We acknowledge NVIDIA for an Academic Hardware Grant.

## References

Sina Ahmadi, John P. McCrae, Sanni Nimb, Fahad Khan, Monica Monachini, Bolette S. Pedersen, Thierry Declerck, Tanja Wissik, Andrea Bellandi, Irene Pisani, Thomas Troelsgård, Sussi Olsen, Simon Krek, Veronika Lipp, Tamás Várad, László Simon, András Gyórfy, Carole Tiberius, Tanneke Schoonheim, Yifat Ben Moshe, Maya Rudich, Raya Abu Ahmad, Dorielle Lonke, Kira Kovalenko, Margit Langemets, Jelena Kallas, Oksana Dereza, Theodorus Fransen, David Cillessen, David Lindemann, Mikel Alonso, Ana Salgado, José Luis Sancho, Rafael-J. Ureña-Ruiz, Kiril Simov, Petya Osenova, Zara Kancheva, Ivaylo Radev, Ranka Stanković, Andrej Perdih, and Dejan Gabrovšek. 2020. A Multilingual Evaluation Dataset for Monolingual Word Sense Alignment. In *Proceedings of the 12th Language Resource and Evaluation Conference (LREC 2020)*, Marseille, France.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal,

- Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press.
- Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. 2019. [Improved word sense disambiguation using pre-trained contextualized word representations](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5297–5306, Hong Kong, China. Association for Computational Linguistics.
- Karin Kipper-Schuler. 2005. *Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania, USA. AAI3179808.
- Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. [An empirical study of incorporating pseudo data into grammatical error correction](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.
- Bajčetić Lenka and Yim Seung-Bin. 2020. Implementation of supervised training approaches for monolingual word sense alignment: Acdh-ch system description for the mwsa shared task at globalex 2020. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, pages 84–91.
- Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. 2020. [SenseBERT: Driving some sense into BERT](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4656–4667, Online. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *Proceedings of the International Conference on Learning Representations (ICLR 2019)*.
- Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. [SemEval-2010 task 14: Word sense induction & disambiguation](#). In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden. Association for Computational Linguistics.
- Raffaele Manna, Giulia Speranza, Maria Pia di Buono, and Johanna Monti. 2020. Unior nlp at mwsa task - globalex 2020: siamese lstm with attention for word sense alignment. In *LREC 2020 Workshop Language Resources and Evaluation Conference 11–16 May 2020*, pages 76–83.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth International Workshop on Semantic Evaluation (SemEval-2021)*.
- Andrea Moro and Roberto Navigli. 2015. [SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking](#). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.
- Roberto Navigli. 2009. [Word sense disambiguation: A survey](#). *ACM Comput. Surv.*, 41(2).
- Roberto Navigli. 2012. [A quick tour of word sense disambiguation, induction and related approaches](#). In *SOFSEM 2012: Theory and Practice of Computer Science*, Lecture Notes in Computer Science, page 115–129. Springer.
- Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. [SemEval-2013 task 12: Multilingual word sense disambiguation](#). In *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzshanskyi. 2020. [GECToR – grammatical error correction: Tag, not rewrite](#). In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.
- Vasile Pais, Dan Tufiş, and Radu Ion. 2020. MWSA task at GlobaLex 2020: RACAI’s word sense alignment system using a similarity measurement of dictionary definitions. In *Proceedings of the 2020 Globalex Workshop on Linked Lexicography*, pages 69–75, Marseille, France. European Language Resources Association.

- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. [WiC: the word-in-context dataset for evaluating context-sensitive meaning representations](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alessandro Raganato, Tommaso Pasini, Jose Camacho-Collados, and Mohammad Taher Pilehvar. 2020. [XL-WiC: A multilingual benchmark for evaluating semantic contextualization](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7193–7206, Online. Association for Computational Linguistics.
- David Strohmaier, Sian Gooding, Shiva Taslimipoor, and Ekaterina Kochmar. 2020. [SeCoDa: Sense complexity dataset](#). In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5962–5967, Marseille, France. European Language Resources Association.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is All you Need](#). In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.
- Loic Vial, Benjamin Lecouteux, and Didier Schwab. 2019. Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. *ArXiv*.
- Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. [Superglue: A stickier benchmark for general-purpose language understanding systems](#). In *Advances in Neural Information Processing Systems*, volume 32, pages 3266–3280. Curran Associates, Inc.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zheng Yuan and Christopher Bryant. 2021. Document-level grammatical error correction. In *Proceedings of the Sixteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.
- Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. [Improving the transformer translation model with document-level context](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542, Brussels, Belgium. Association for Computational Linguistics.

# UoB\_UK at SemEval 2021 Task 2: Zero-Shot and Few-Shot Learning for Multi-lingual and Cross-lingual Word Sense Disambiguation.

Wei Li, Harish Tayyar Madabushi and Mark Lee

School of Computer Science  
University of Birmingham  
United Kingdom

WXL885@student.bham.ac.uk

Harish@HarishTayyarMadabushi.com, M.G.Lee@bham.ac.uk

## Abstract

This paper describes our submission to SemEval 2021 Task 2. We compare XLM-RoBERTa Base and Large in the few-shot and zero-shot settings and additionally test the effectiveness of using a k-nearest neighbors classifier in the few-shot setting instead of the more traditional multi-layered perceptron. Our experiments on both the multi-lingual and cross-lingual data show that XLM-RoBERTa Large, unlike the Base version, seems to be able to more effectively transfer learning in a few-shot setting and that the k-nearest neighbors classifier is indeed a more powerful classifier than a multi-layered perceptron when used in few-shot learning.

## 1 Introduction and Motivation

Word Sense Disambiguation (WSD) is the task of disambiguating semantic meaning at the word level and is an important part of Natural Language Processing (NLP) with applications in several downstream tasks (Wang et al., 2020b). In recent years, Few Shot Learning (FSL) has been successful in several domains (Wang et al., 2020a) including in NLP (Yan et al., 2018). Modern deep neural networks require a significant amount of training data that might not always be available. FSL is a solution to this problem, wherein training data from a related domain or language can be used to augment training on the target domain/language with significantly less data. FSL can be characterised as n-way k-shot classification.

We participate in SemEval Task 2 Multilingual and Cross-lingual Word in Context Disambiguation (Martelli et al., 2021), which provides 8,000 training examples in English but only 32 in the other target languages. In addition the tasks requires disambiguation of word pairs in the cross lingual setting between EN-AR, EN-FR, EN-RU,

and EN-ZH, where only 8 examples each are available. This sparsity in training data led us to explore the use of FSL for this task.

We hypothesise that:

1. “Large” models, which have been shown to have a lot more linguistic information and high-level generalisability, are more likely to be able to generalise their learning in the few-shot setting, and
2. that the use of a k-nearest neighbors (KNN) classifier is likely to be more effective in the few-shot setting. This hypothesis is based on our exploration of related work (Section 2).

Our experiments on both the multi-lingual and cross-lingual data show that these hypotheses are in fact correct. We show that XLM-RoBERTa Large, unlike the Base version, seems to be able to more effectively transfer knowledge in a few-shot setting, and that the KNN classifier is indeed a more powerful classifier than a multi-layered perceptron (MLP) when used in few-shot learning. To ensure reproducibility and so other researchers can build on this work, we release the program code, hyperparameters and experiments associated with this work<sup>1</sup>.

## 2 Related work

The first effective method that implemented Few Shot Learning in NLP was that by Koch et al. (2015), who introduced the application of the siamese network in one-shot learning. The siamese network is typically used to calculate semantic similarity between sentences and was shown to be powerful in the FSL setting.

A recent use of zero-shot learning in WSD was work by Kumar et al. (2019), who proposed the

<sup>1</sup><https://github.com/weilk/SemEval-2021-Task-2>

extension of WSD systems by incorporating Sense embeddings (EWISE). These sense embeddings are derived from a knowledge graph, namely WordNet (Miller, 1998), and graph embeddings. EWISE predicts over an embedding space instead of the discrete label space and allow generalized zero-shot learning capability. Instead of using annotated data, it uses definitions from WordNet.

Pelevina et al. (2017) proposed a simple and effective method which uses clustering in ego-networks. Egocentric networks are local networks with one central node, known as the “ego”. This method allows labeling of words in the context with learned sense vectors thus providing a new approach to unsupervised WSD.

Our use of the K Nearest Neighbours (KNN) classifier is motivated by the work by Snell et al. (2017), who proposed a very straightforward network, similar to the nearest class mean approach (Mensink et al., 2013), that makes use of the classes of “prototypical” examples to classify new examples based on their distances. We extend this method by use of pre-trained models and a KNN classifier (Section 3).

### 3 Methodology

In this section, we describe the different models used for the task. Our models are built on top of XLM-RoBERTa, but because cross-lingual data is limited, we used FSL during training. We also perform data preprocessing, especially in the case of Chinese and Arabic where tokenisation is inexact.

#### 3.1 Data Preprocessing

We only use data available from the official data set provided<sup>2</sup> for all our experiments. We performed additional preprocessing in the case of Chinese and Arabic as it is often difficult to locate the target word in these cases. This is because Conneau et al. (2020) use the SentencePiece algorithm as the basis of XLMRobertaTokenizer, which tends to output the largest granularity words by meaning in both Chinese and Arabic. Due to this, it is possible that the target word may either be included in the hypernym’s word-piece or be cut and included in a different hypernym’s word-piece.

For example, XLM-RoBERTa tokeniser was used to tokenise a Chinese sentence, and the output is shown in Figure 1. The target word is “事情”.

<sup>2</sup><https://github.com/SapienzaNLP/mcl-wic>

However, the tokeniser includes the target word in the larger word “这件事情”. To get around this and to ensure correct word tokenisation, we add a comma (“,”) around the target word in both Chinese and Arabic sentences as in: “中国报告这件事情,”.

```
In [22]: tokenizer.tokenize("中国报告这件事情")
Out[22]: ['_中国', '报告', '这件事情']
```

Figure 1: Tokenise Chinese sentence

### 3.2 System Architecture

In order to find the most effective model architecture, we experiment with different variations: The cosine similarity between contextual word representations obtained from XLM-RoBERTa (Conneau et al., 2020) using multiple thresholds, classification of pre-generated XLM-RoBERTa embeddings using a multi-layer perceptron, and finally an end to end model with a softmax layer for classification. Our experiments showed the end to end model to be most effective, which we use for all downstream experimentation.

#### 3.2.1 The Base End-to-end Model

Models used in this work are variations of the base end-to-end model detailed in Figure 2. The model takes as input the two sentences and the positions of the target words. Each of the input sentences are transformed into the contextual word embeddings using XLM-RoBERTa and from the resultant embeddings the contextual embeddings associated with the target word are selected. These vectors,  $v_1$  and  $v_2$  are further augmented with their difference and concatenated into a single vector  $v_1, v_2, v_1 - v_2, v_2 - v_1$ . We note that although adding the vectors  $v_1 - v_2$  and  $v_2 - v_1$  does not provide the model with any additional information, our initial experiments showed that this boosted performance.

This combined vector is then passed through a transformer layer and a mean pool layer splits the output of the transformer layer into two different vectors which are compared using cosine similarity. The cosine similarity of these vectors is used to build a vector  $(cos, 1 - cos)$  which is then passed through either an MLP or a KNN classifier. This output is finally passed through a softmax layer to classify the word in the two sentences as belonging to the same meaning or not.

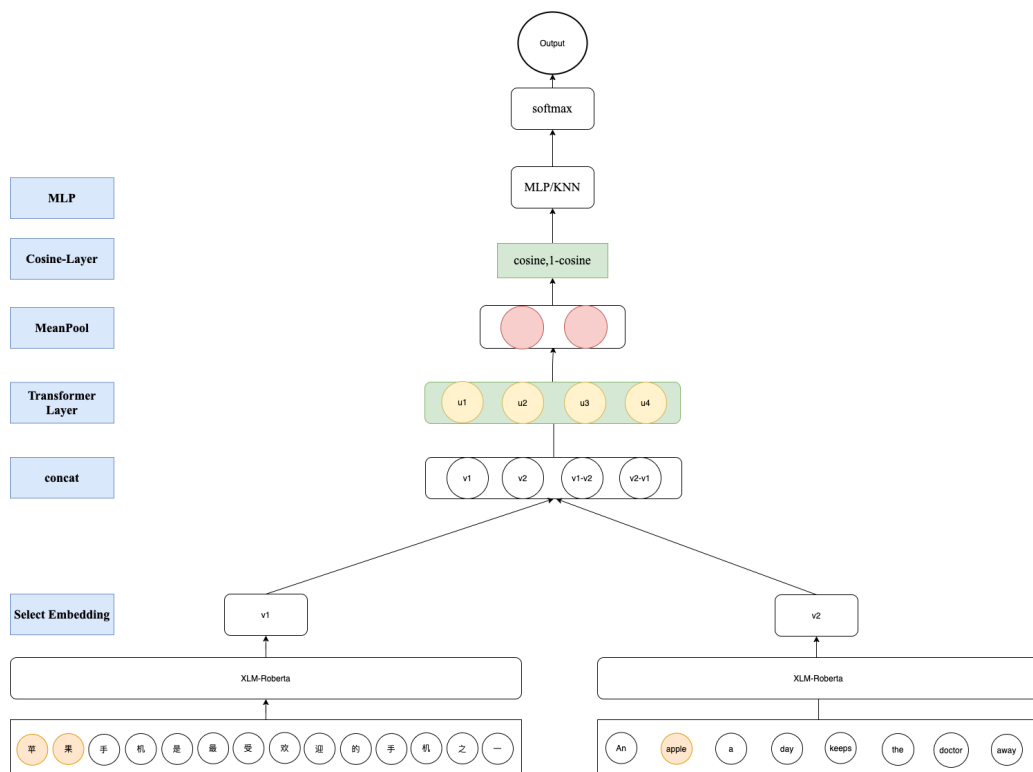


Figure 2: The architecture of the model used for sense disambiguation.

### 3.2.2 Zero-Shot Models

The zero-shot methods we use are variations of the base model, one which uses the Base version of XLM-RoBERTa, and the other which uses the Large version of XLM-RoBERTa. In this setting we use an MLP classifier instead of the KNN classifier. These models are trained on the English training data and tested on the multi-lingual and cross-lingual settings.

### 3.2.3 Few-Shot Models

The models that we use in the few-shot setting are modifications of the zero-shot model and we use the zero-shot model as a baseline. In all cases we start with the model trained on the English training data. The first variation replaces the multi-layered perceptron with a KNN classifier (with  $k = 2$ ), and the second variation replaces XLM-RoBERTa Base with XLM-RoBERTa Large. These variations are further detailed in Section 4 and are trained on the minimal data available in the target language pairs.

## 4 Experiments Design

We design several experiments to find the best performing model. All the experiments are performed

on the same platform<sup>3</sup>. We also use warmup in all the experiments and further hyperparameters are detailed in the documentation associated with the program code released with this work. For each experiment, five different random seed are tested and we choose the best performing model.

We use the English training data to train our baseline and zero-shot models. The development data provided consisted of 1000 examples for each language in the multi-lingual setting (en-en, ar-ar, zh-zh, fr-fr, and ru-ru). We divide this into a Dev-Train subset consisting of 600 examples, a Dev-Validation and Dev-Test subsets consisting of 200 examples each. In the cross-lingual setting, where we have only 8 examples from each language pair, we use the trial data for few shot training and do not use a validation or test set due to data sparsity. Finally, due to data sparsity in the target languages we freeze the transformer layers of XLM-RoBERTa during the few shot training phase.

Based on the results of these experiments, we select the best five models to submit to the official websites. These models were:

<sup>3</sup>System: Ubuntu 16.04.6 LTS. CPU:Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz. 4 cores,8 threads. GPU:GeForce RTX 3090 24GB. Pytorch+cuda:1.7.1+cu110. Python 3.7.4



	EN-AR	EN-FR	EN-RU	EN-ZH
XLM-RoBERTa Base + MLP, Zero-Shot	74.30 (46)	80.00 (39)	81.60 (35)	76.30 (43)
XLM-RoBERTa Large + MLP, Zero-Shot	76.70 (41)	<b>84.00 (19)</b>	82.90 (28)	81.00 (37)
XLM-RoBERTa Base + MLP, Few-shot	73.00 (49)	76.50 (50)	80.10 (40)	75.50 (44)
XLM-RoBERTa Large + MLP, Few-shot	80.40 (34)	81.40 (34)	80.70 (38)	81.80 (33)
XLM-RoBERTa Large + KNN, Few-shot	<b>81.90 (30)</b>	83.90 (20)	<b>83.30 (24)</b>	<b>83.60 (29)</b>

Table 1: Accuracy on the final cross-lingual test set with the rank achieved by that submission in brackets. The highest score for each language pair is highlighted in bold. Rows 1, 3 and 2, 4 are comparable results between zero-shot and few-shot settings.

	EN-EN	AR-AR	FR-FR	RU-RU	ZH-ZH
XLM-RoBERTa Base + MLP, Zero-Shot	84.50 (50)	78.20 (40)	78.60 (44)	78.10 (34)	81.40 (32)
XLM-RoBERTa Large + MLP, Zero-Shot	87.30 (37)	77.30 (43)	<b>84.20 (18)</b>	<b>82.30 (23)</b>	80.80 (35)
XLM-RoBERTa Base + MLP, Few-shot	84.40 (51)	78.90 (36)	79.20 (41)	78.10 (34)	80.60 (36)
XLM-RoBERTa Large + MLP, Few-shot	87.10 (38)	<b>81.00 (27)</b>	83.40 (22)	82.00 (24)	82.00 (28)
XLM-RoBERTa Large + KNN, Few-shot	<b>88.50 (33)</b>	78.40 (38)	83.60 (21)	81.90 (25)	<b>82.10 (27)</b>

Table 2: Accuracy on the final multi-lingual test set with the rank achieved by that submission in brackets. The highest score for each language pair is highlighted in bold. Rows 1, 3 and 2, 4 are comparable results between Zero-Shot and few shot settings. The variation in en-en is a result of random initialisation.

1. XLM-RoBERTa Base with a multi-layered perceptron as the **baseline** zero-shot model.
2. XLM-RoBERTa Large with a multi-layered perceptron as a second zero-shot model.
3. XLM-RoBERTa Base with a multi-layered perceptron, additionally trained on available target language data as a few-shot model.
4. XLM-RoBERTa Large with a multi-layered perceptron, additionally trained on available target language data as a second few-shot model.
5. XLM-RoBERTa Large with a K-Nearest Neighbour classifier, additionally trained on available target language data as a third few-shot model.

## 5 Results and Analysis

The final results in the cross-lingual and multi-lingual settings are displayed in Tables 1 and 2 respectively. Each table displays the accuracy on the corresponding test set with the rank achieved by that submission in brackets (out of a total of 87 teams).

In each of the two tables, rows 1, 3 and 2, 4 provide a comparison between the zero-shot and few shot settings. It is interesting to note that few shot learning is *not* effective when using the base

version of XLM-RoBERTa. Across both the cross-lingual and multi-lingual settings, the minimal additional data does little to boost performance and in some cases actually reduces performance.

XLM-RoBERTa Large on the other hand, seems to be able to transfer knowledge extracted from training in English to the other languages and *gains the most when those languages are significantly different from English, the language in which the majority of the training data is available in*. The impact of few shot learning is the largest when the difference between the original training data (in this case English) and the target languages is largest. As can be seen from Table 1, the increase on the English-Arabic test set between the zero-shot and few shot settings is *nearly 5 percentage points* despite the few-shot model having been trained on only 8 examples. This same trend can be observed on the English-Chinese dataset albeit to a smaller extent.

The use of a KNN classifier, in place of an MLP, improves performance across the board providing the best results in a lot of the cases and comparable results in the rest. These results seem to validate the results obtained by Snell et al. (2017), who show that the use of a KNN to classify examples related to prototypical examples is an effective method in few-shot learning (Section 2).

## 6 Conclusions and Future work

This paper described our submission to SemEval 2021 Task2, multi-lingual and cross-lingual word in context disambiguation. Given the nature of the task, wherein we are provided with training data in English and very limited training data in the other target languages, we use zero-shot and few-shot learning.

We hypothesised (Section 1) that two methods will significantly boost performance in the few-shot learning setting: a) The use of “Large” pre-trained models which have been shown to have access to a lot more linguistic information and so generalisability, and b) the use of a KNN classifier instead of a multi-layered perceptron.

Our experiments, described in Section 5, confirm that this is indeed the case. We find that XML-RoBERTa Large is able to significantly increase performance in the few-shot setting, especially when the target languages are dissimilar to English, which is the language the majority of the training data is available in. We additionally find that the use of a KNN classifier boosts performance in the few-shot setting.

We additionally show that when using pre-trained models, tokenisers might split words in ways that are not conducive to the task at hand, especially in languages such as Chinese, where word delimitation is inexact. We handle this limitation by using a comma to delimit words in ways that are specific to the problem, which is both effective and easy to implement.

In future, we intend to explore the use of these methods on other datasets and problems beyond word sense disambiguation.

## References

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. [Zero-shot word sense dis-](#)

[ambiguation using sense definition embeddings](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy. Association for Computational Linguistics.

- F. Martelli, N. Kalach, G. Tola, and R. Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the 15th Workshop on Semantic Evaluation, 2021*.
- T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. 2013. [Distance-based image classification: Generalizing to new classes at near-zero cost](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637.
- George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.
- Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. [Making sense of word embeddings](#). *CoRR*, abs/1708.03390.
- Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. [Prototypical networks for few-shot learning](#). *CoRR*, abs/1703.05175.
- Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. 2020a. [Generalizing from a few examples: A survey on few-shot learning](#). *ACM Comput. Surv.*, 53(3).
- Yinglin Wang, Ming Wang, and Hamido Fujita. 2020b. [Word sense disambiguation: A comprehensive knowledge exploitation framework](#). *Knowledge-Based Systems*, 190:105030.
- Leiming Yan, Yuhui Zheng, and Jie Cao. 2018. Few-shot learning for short text classification. *Multimedia Tools and Applications*, 77(22):29799–29810.

# PAW at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation : Exploring Cross Lingual Transfer, Augmentations and Adversarial Training

Harsh Goyal, Aadarsh Singh and Priyanshu Kumar

Indian Institute of Technology (Indian School of Mines) Dhanbad, India  
{harshgoyal13, aadarshsingh191198, kpriyanshu256}@gmail.com

## Abstract

We experiment with XLM RoBERTa for Word in Context Disambiguation in the Multi Lingual and Cross Lingual setting so as to develop a single model having knowledge about both settings. We solve the problem as a binary classification problem and also experiment with data augmentation and adversarial training techniques. In addition, we also experiment with a 2-stage training technique. Our approaches prove to be beneficial for better performance and robustness.

## 1 Introduction

Language is complex even for human beings, let alone for computers. The same word serves different purposes in different scenarios, thus increasing the complexity of the Word Sense Disambiguation (WSD). For example in English, the word “bank” can refer to a financial institution or the land alongside a river. Many works revolving around WSD have been done with the help of explicit word sense inventories like WordNet<sup>1</sup> and BabelNet<sup>2</sup>. With the advent of advanced deep learning models, it is desirable to develop systems that have a good understanding of languages without such gold standards of word sense. This unsupervised learning can help the model learn better latent representations of words in different contexts.

In this paper, we aim to develop a single system that has knowledge of both multilingual and cross-lingual word sense disambiguation by training models with the combined data for both settings. We present our approaches for WSD in the multilingual and cross-lingual domain. The task is treated as a binary-classification problem: whether words have the same sense in the two given pairs of sentences. We experiment with XLM-RoBERTa

(Conneau et al., 2019), which is based on the Transformer architecture (Vaswani et al., 2017), as the backbone of our architectures in both the settings. In addition, we also leverage external data and different training techniques and data augmentation.

The rest of the paper is organized as follows : various related works have been discussed in section 2, followed by a brief description of the shared task dataset in section 3. The system overview and experimental settings are covered in sections 4 and 5. Sections 6 contain the results. Section 7 concludes the paper and also includes scope of future work.

## 2 Related Work

Silberer and Ponzetto (2010) make use of graph algorithms for the word sense disambiguation task. They build a multilingual co-occurrence graph in which the multilingual nodes are connected with translation edges and labelled with the target word’s translations as obtained from the corresponding contexts.

Authors in Banea and Mihalcea (2011), use multilingual vector space which is obtained by expanding monolingual features engineered from more than one language, in order to generate a more effective, robust and utilitarian vector representation. These engineered features are then used for WSD.

Languages like Arabic do not have as many resources in the available dataset as compared to more common languages like English. To tackle this issue for the Persian language, Lefever and Hoste (2011) follow a two phase approach - in the first phase, they utilize an English Word sense disambiguation system to assign “sense tags” to words appearing in English sentences and then in the following phase, they transfer the senses obtained in the previous phase to corresponding Persian words.

In the Semeval-2013 WSD task (Navigli et al.,

<sup>1</sup><https://wordnet.princeton.edu/>

<sup>2</sup><https://babelnet.org/>

2013), Rudnick et al. (2013) take a classification-based approach to the Cross-Lingual WSD task. They build the HLTDI system in which they perform word alignment on the Europarl corpus. This helps them find samples in the training data which have ambiguous focus words. The paper describes three variants of the classifier - one is trained over local features, the second is trained over the data with translation of the focus word in the four target languages added to the feature vector and the final variant builds a Markov network of the first classifier in order to find the best translation.

A few works have also been submitted as a part of SemDeep-5 workshop (Espinosa-Anke et al., 2019). Ansell et al. (2019) make use of contextualised ELMo word embeddings. A Bidirectional Long Short Term Memory(LSTM) cell is used to extract better representation of the given sentences. To disambiguate the words, they optimise the cosine distance between the concatenated hidden representations of words, preceding and following the target word. Soler et al. (2019) augment the dataset by automatically substituting target words using contextual similarity. They then experiment with different contextual word embeddings and train a logistic regression classifier on top of that.

### 3 Dataset

The dataset (Martelli et al., 2021)<sup>3</sup> provided by the shared task organizers consists of both multilingual and cross-lingual data in English (EN), Arabic (AR), French (FR), Russian (RU) and Chinese (ZH). The dataset consists of two sentences and the words in corresponding sentences (which need disambiguation) and the corresponding label.

### 4 System Overview

Our experiments revolve around Facebook’s XLM RoBERTa model, which was an update to their XLM-100 Language Model. XLM RoBERTa is based on the transformer architecture consisting of multi-attention heads which apply a sequence-to-sequence transformation on the input text sequence. The training procedure is inspired from RoBERTa (Liu et al., 2019) i.e. only the Masked Language Model objective is used. XLM RoBERTa is scaled up to 100 languages, thus becoming a good choice for multi-lingual datasets.

Another motivation to experiment with XLM RoBERTa comes from the facility of “Cross Lin-

gual Transfer”, which can help with unbalanced data of different languages. Knowledge is transferred for all languages if the model is trained for a particular task using data of only one language. Thus, this feature saves effort of gathering more data to make the data distribution balanced.

#### 4.1 Problem Formulation

We perform experiments keeping the model architecture constant across all experiments. The model accepts both the sentences concatenated together. The input to the model is formulated as :  $word_1 + \langle /s \rangle + sentence_1 + \langle /s \rangle + word_2 + \langle /s \rangle + sentence_2$ , where  $\langle /s \rangle$  is the separator token in XLM RoBERTa vocabulary.

Dropout is applied on the pooled encoding output from the model. The dropout probability is set to 0.3. The dropout applied output is then passed through a linear layer which provides us with the logits corresponding to the 2 classes.

#### 4.2 Data Augmentation

Data augmentations are considered an important technique to avoid overfitting of neural networks thus making them more generalised. Since our model architecture accepts both the sentences together, there is room to apply a simple data augmentation during training. Consider  $t_1$  and  $t_2$  are the 2 sentences for a particular data instance. The training data is augmented as  $t_1 \oplus t_2$  and  $t_2 \oplus t_1$ , where  $\oplus$  represents concatenation. We apply the augmentation taking care that no data leakage takes place in the validation data.

#### 4.3 Two Stage Training

To leverage the property of Cross Lingual Transfer, we first train the model on the WiC dataset (Pilehvar and Camacho-Collados, 2018), which consists only of English data. Then we train the same model (trained on WiC) on the MCL WiC dataset. This technique instills some knowledge via cross lingual transfer, about WSD in the first stage and then builds on the knowledge using the shared task dataset.

#### 4.4 Adversarial Training (AT)

Adversarial training is another technique that is used to increase the robustness of models, which also helps in better generalisation. Adversarial training in Computer Vision is done by directly perturbing the input images. However, text data

<sup>3</sup><https://github.com/SapienzaNLP/mcl-wic>

Model	EN-AR	EN-FR	EN-RU	EN-ZH
2 stage + Train Aug	75.3	78.2	78.7	75.2
2 stage + Train Aug + TTA	74.6	78.0	<b>78.9</b>	<b>75.3</b>
2 stage + Train Aug + AT	76.8	78.1	76.9	74.6
2 stage + Train Aug + AT + TTA	<b>76.9</b>	<b>79.5</b>	78.0	74.6
Best performance	89.1	89.1	89.4	91.2

Table 2: Test Scores for Cross Lingual Setting

being discrete in nature, the perturbations are added to the word embeddings.

Many approaches for adversarial training in NLP have been developed. We experiment using Miyato et al. (2016) approach with little modification. In their approach, the word embeddings are normalized first. Required perturbations are created using the gradients obtained via backpropagation. Let the sequence of (normalized) word embedding vectors of a text be  $t$ . The model parameters are represented by  $\theta$ . The probability of the text belonging to class  $y$  is given by  $p(y|t; \theta)$ . The adversarial perturbations  $z_{adv}$  are computed as follows:

$$g = \nabla_t \log p(y|t; \theta)$$

$$z_{adv} = -\epsilon g / \|g\|_2$$

where  $\epsilon$  is a hyper-parameter controlling the size of the perturbations. The adversarial loss is defined as :

$$L_{adv}(\theta) = -\frac{1}{N} \sum_{n=1}^N \log p(y_n | t_n + z_{adv,n}; \theta)$$

By using the gradients calculated from the above loss, the weights of the model are updated (the non-perturbed word embeddings of the model are updated). Our experiments deviate from the above method in the part that we do not normalize our pre-trained word embedding of the model, since doing so might change the semantic meaning of the pre-trained word embeddings. We perform adversarial training XLM RoBERTa model using  $\epsilon = 1$ .

Model	CV
W/O extra techniques	73.68
Train Aug	74.68
2 stage + Train Aug	75.64
2 stage + Train Aug + AT	<b>77.07</b>

Table 1: Cross Validation Scores

#### 4.5 Test Time Augmentation (TTA)

The usage of the training data augmentation can be extended to test time as well. For a given data instance  $t_1$  and  $t_2$ , the model predictions for  $t_1 \oplus t_2$  and  $t_2 \oplus t_1$  are combined using simple averaging of probabilities. Thus, this simple augmentation can help boost the performance of the model.

## 5 Experimental Setup

We make use of combined training and validation data provided by the shared task organizers. We perform a stratified 5 fold cross validation using the combined data. In all our experiments, we fine tune the entire model. Each fold is trained for 20 epochs using early stopping with patience of 6 and tolerance of  $1e-3$ . The models are optimised using AdamW (Loshchilov and Hutter, 2017) with a learning rate of  $5e-6$  and a batch size of  $16^4$ . Inputs of maximum sequence length 172 are used in the model. The models have been implemented using Pytorch (Paszke et al., 2019) and Huggingface’s Transformers (Wolf et al., 2019) library.

## 6 Results

Accuracy score is the official evaluation metric for the shared task. The test predictions are obtained by combining the predictions of all the 5 fold models (by averaging the predictions from all models). Table 1 lists down the cross validation accuracy scores of all the experiments. The test scores are categorised as cross-lingual and multilingual and are presented in tables 2 and 3 respectively. For bench marking purpose, we also mention the best performances achieved by participants of the shared task.

A few observations can be made by looking at the results:

<sup>4</sup>It is important to note that XLM RoBERTa requires a much smaller learning rate as compared to BERT and other models, for training; XLM RoBERTa is incapable of learning if trained using high learning rates like  $2e-5$

Model	EN-EN	AR-AR	FR-FR	RU-RU	ZH-ZH
2 stage + Train Aug	84.5	79.7	78.8	77.4	79.6
2 stage + Train Aug + TTA	<b>85.4</b>	<b>80.2</b>	79.0	<b>77.7</b>	<b>80.4</b>
2 stage + Train Aug + AT	85.2	79.0	80.2	76.9	79.2
2 stage + Train Aug + AT + TTA	85.1	80.0	<b>80.5</b>	77.6	79.7
Best performance	93.3	84.8	87.5	87.4	91.0

Table 3: Test Scores for Multilingual Setting

1. Test Time Augmentation helps in boosting the scores.
2. In the cross lingual setting, models trained with and without adversarial training are competent to the same extent. On the other hand, in the multilingual setting, models trained without adversarial training seem to have the upper hand.

## 7 Conclusion and Future Work

We explore the performance of XLM RoBERTa at Word In Context Disambiguation both in the multilingual and cross lingual setting. We also explore different training techniques such as two-stage training and adversarial training along with some simple augmentations to make our models robust and more generalized. Test Time Augmentations, based on training augmentation turn out to be useful. For future work, we can explore the performance of ensembling different kinds of models trained with and without adversarial training together, so as to produce more robust results. It will also be interesting to experiment with larger backbone models in the current architecture.

## Acknowledgments

We thank Kaggle for providing free GPU and TPU services.

## References

Alan Ansell, Felipe Bravo-Marquez, and Bernhard Pfahringer. 2019. An ELMo-inspired approach to semdeep-5’s word-in-context task. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 21–25.

Carmen Banea and Rada Mihalcea. 2011. Word sense disambiguation with multilingual features. In *Proceedings of the Ninth International Conference on Computational Semantics (IWCS 2011)*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Luis Espinosa-Anke, Thierry Declerck, Dagmar Gromann, Jose Camacho-Collados, and Mohammad Taher Pilehvar, editors. 2019. *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*. Association for Computational Linguistics, Macau, China.

Els Lefever and Veronique Hoste. 2011. Examining the validity of cross-lingual word sense disambiguation. *Polibits*, (43):29–35.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. [Fixing weight decay regularization in adam](#). *CoRR*, abs/1711.05101.

Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013. [SemEval-2013 task 12: Multilingual word sense disambiguation](#). In *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca

- Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.
- Alex Rudnick, Can Liu, and Michael Gasser. 2013. Hltdi: Cl-wsd using markov random fields for semeval-2013 task 10. In *Second Joint Conference on Lexical and Computational Semantics (\* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 171–177.
- Carina Silberer and Simone Paolo Ponzetto. 2010. UHD: Cross-lingual word sense disambiguation using multilingual co-occurrence graphs. In *Erk K, Strapparava C, editors. Proceedings of the 5th International Workshop on Semantic Evaluation; 2010 Jul 15-16; Uppsala, Sweden. Stroudsburg: ACL; 2010. p. 134-7. ACL (Association for Computational Linguistics)*.
- Aina Garí Soler, Marianna Apidianaki, and Alexandre Allauzen. 2019. LIMSI-MULTISEM at the IJCAI Semdeep-5 Wic challenge: Context representations for word usage similarity estimation. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 6–11.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s Transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.

# LU-BZU at SemEval-2021 Task 2: Word2Vec and Lemma2Vec performance in Arabic Word-in-Context disambiguation

**Moustafa Al-Hajj**  
Lebanese University  
Lebanon

moustafa.alhajj@ul.edu.lb

**Mustafa Jarrar**  
Birzeit University  
Palestine

mjarrar@birzeit.edu

## Abstract

This paper presents a set of experiments to evaluate and compare between the performance of using CBOV Word2Vec and Lemma2Vec models for Arabic Word-in-Context (WiC) disambiguation without using sense inventories or sense embeddings. As part of the SemEval-2021 Shared Task 2 on WiC disambiguation, we used the `dev.ar-ar` dataset (2k sentence pairs) to decide whether two words in a given sentence pair carry the same meaning. We used two Word2Vec models: Wiki-CBOV, a pre-trained model on Arabic Wikipedia, and another model we trained on large Arabic corpora of about 3 billion tokens. Two Lemma2Vec models was also constructed based on the two Word2Vec models. Each of the four models was then used in the WiC disambiguation task, and then evaluated on the SemEval-2021 `test.ar-ar` dataset. At the end, we reported the performance of different models and compared between using lemma-based and word-based models.

## 1 Introduction

As a word may denote multiple meanings (*i.e.*, senses) in different contexts, disambiguating them is important for many NLP applications, such as information retrieval, machine translation, summarization, among others. For example, the word “table” in sentences like “I am cleaning the table”, “I am serving the table”, “I am emailing the table”, refer to “furniture”, “people”, and “data” respectively. Disambiguating the sense that a word denotes in a given sentence is important for understanding the semantics of this sentence.

To automatically disambiguate word senses in a given context, many approaches have been proposed based on supervised, semi-supervised, or unsupervised learning models. Supervised and semi-supervised methods rely on full, or partial, labeling of the word senses in the training corpus

to construct a model (Lee and Ng, 2002; Klein et al., 2002). On the other hand, unsupervised approaches induce senses from unannotated raw corpora and do not use lexical resources. The problem in such approaches, is that unsupervised learning of word embeddings produces a single vector for each word in all contexts, and thus ignoring its polysemy. Such approaches are called static Word Embeddings. To overcome the problem, two types of approaches are suggested (Pilehvar and Camacho-Collados, 2018): multi-prototype embeddings, and contextualized word embeddings. The latter suggests to model context embeddings as a dynamic contextualized word representation in order to represent complex characteristics of word use. Proposed architectures such as ELMo (Peters et al., 2018), ULMFiT (Howard and Ruder, 2018), GPT (Radford et al., 2018), T5 (Raffel et al., 2019), and BERT (Devlin et al., 2018), achieved breakthrough performance on a wide range of natural language processing tasks. In multi-prototype embeddings, a set of embedding vectors are computed for each word, representing its senses. In (Pelevina et al., 2017), multi-prototype embeddings are produced based on the embeddings of a word. As such, a graph of similar words is constructed, then similar words are grouped into multiple clusters, each cluster representing a sense. As for Mancini et al. (2016), multi-prototype embeddings are produced by learning word and sense embeddings jointly from both, a corpus and a semantic network. In this paper we aim at using static word embeddings for WiC disambiguation.

Works on Arabic Word Sense Disambiguation (WSD) are limited, and the proposed approaches are lacking a decent or common evaluation framework. Additionally, there are some specificities of the Arabic language that might not be known in other languages. Although polysemy and disambiguating are challenging issues in all languages,



they might be more challenging in the case of Arabic (Jarrar et al., 2018; Jarrar, 2021) and this for many reasons. For example, the word *šāhd* (شاهد) could be *šāhid* (شاهد) which means a *witness*, or *šāhada* (شاهد) which means *watch*. As such, disambiguating words senses in Arabic, is similar to disambiguating senses of English words written without vowels. Second, Arabic is a highly inflected and derivational language. As such, thousands of different word forms could be inflected and derived from the same stem. Therefore, words in word embeddings models will be considered as different, which may affect the accuracy and the utility of their representation vectors, as the same meaning could be incarnated in distributed word forms in corpora, which has led some researchers to think that using lemma-based models might be better than using word-based embeddings in Arabic (Salama et al., 2018; Shapiro and Duh, 2018). This idea will be discussed later in sections 5 and 6.

Alkhatlan et al. (2018) suggested an Arabic WSD approach based on Stem2Vec and Sense2Vec. The Stem2Vec is produced by training word embeddings after stemming a corpus, whereas the Sense2Vec is produced based on the Arabic WordNet sense inventory, such that each synset is represented by a vector. To determine the sense of a given word in a sentence, the sentence vector is compared with every sense vector, then the sense with maximum similarity is selected.

Laatar et al. (2017) did not use either stemming or lemmatization. Instead, they proposed to determine the sense of a word in context by comparing the context vector with a set of sense vectors, then the vector with the maximum similarity is selected. The context vector is computed as the sum of vectors of all words in a given context, which are learnt from a corpus of historical Arabic. On the other hand, sense vectors are produced based on dictionary glosses. Each sense vector is computed as the sum of vectors (learnt from the historical Arabic corpus) of all words in the gloss of a word.

Other approaches to Arabic WSD (Elayeb, 2019) employ other techniques in machine learning and knowledge-based methods (Bouhriz et al., 2016; Bousmaha et al., 2013; Soudani et al., 2014; Merhbene et al., 2014; Al-Maghasbeh and Bin Hamzah, 2015; Bounhas et al., 2015).

In this paper, we present a set of experiments to evaluate the performance of using Lemma2Vec vs CBOV Word2Vec in Arabic WiC disambiguation.

The paper is structured as follows: Section 2 presents the background of this work. Section 3 overviews the WiC disambiguation system. Section 4 and Section 5, respectively, present the Word2Vec and Lemma2Vec models. In Section 6 we present the experiments and the results; and in section 7 we summarize our conclusions and future work.

## 2 Background

Experiments presented in this paper are part of the SemEval shared task for Word-in-Context disambiguation (Martelli et al., 2021).

The task aims at capturing the polysemous nature of words without relying on a fixed sense inventory. A common evaluation dataset is provided to participants in five languages, including Arabic, our target language in this paper. The dataset was carefully designed to include all parts of speeches and to cover many domains and genres. The Arabic dataset (called multilingual *ar-ar*) consists of two sets: a train set of 1000 sentence pairs for which tags (TRUE or FALSE) are revealed, and a test set of 1000 sentence pairs for which tags were kept hidden during the competition. Figure 1 gives two examples of sentence pairs in the *dev.ar-ar* dataset. Each sentence pair has a word in common for which start and end positions in sentences are provided. Participants in the shared task were asked to infer whether the target word carries the same meaning (TRUE) or not (FALSE) in the two sentences.

```
{
  "id": "dev.ar-ar.0",
  "lemma": "ملاك",
  "pos": "NOUN",
  "sentence1": "ونظرا لأهمية هذه المسائل لسير عمل المحكمة",
  "sentence2": "مستقبلا، يلزم توفير ملاك كاف من الموظفين منذ بدء عملياتها ولا توجد حراسة أمام جميع البعثات الدبلوماسية".
}
{
  "id": "dev.ar-ar.1",
  "lemma": "ملاك",
  "pos": "NOUN",
  "sentence1": "ونظرا لأهمية هذه المسائل لسير عمل المحكمة",
  "sentence2": "مستقبلا، يلزم توفير ملاك كاف من الموظفين منذ بدء عملياتها وأُعربت عن رغبتها في الحصول على معلومات بشأن موارد هاتين الوزارتين وملاكهما وبشأن لجان إعادة التأهيل وإعادة التوطين والمساعدة الوارد ذكرها في العرض الشفوي".
}
```

Figure 1: Two examples of sentence pairs.

### 3 System Overview

This section describes our method to Arabic WiC disambiguation based on two types of embeddings: CBOV Word2Vec and Lemma2Vec.

Given two sentences,  $s_1$  and  $s_2$ , and two words,  $v_i$  from  $s_1$  and  $w_j$  from  $s_2$ , the objective is to check whether  $v_i$  and  $w_j$  have the same meaning. To this end, our system extracts contexts of  $v_i$  and  $w_j$  from the sentence pair, represents them in two vectors and finally compares the two resulting vectors using the cosine similarity. The context of a word  $w$  of size  $n$  (denoted by  $context(w, n)$ ) is composed of the words that surround the word  $w$ , with  $n$  words on the left and  $n$  words on the right ( $n$  varying between 1 and 10 in conducted experiments). To represent  $context(w, n)$  in a vector space, two methods are proposed: first one is based on CBOV Word2Vec embeddings vectors (Mikolov et al., 2013) of the words appearing in the context, whereas the second is based on the Lemma2Vec of lemmas of words appearing in the context. To select the best way to represent the  $context(w, n)$  by a vector, classification experiments were conducted using (i) different pooling operations, *min*, *max*, *mean*, and *std* to combine words/lemmas vectors of the context, (ii) different threshold values (between 0.55 and 0.85) and (iii) the removal of functional words (also called stop words). The later are used to express grammatical relationships among other words, they are characterized by they high frequency in the corpus which might affect the WiC disambiguation accuracy. The cosine similarity is then used to compare vectors of  $context(v_i, n)$  and  $context(w_j, n)$ . Figure 2 illustrates how the cosine similarity is calculated from  $context(v_i, 3)$  and  $context(w_j, 3)$ .

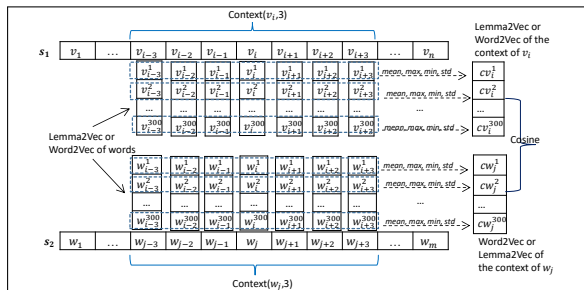


Figure 2: Calculation of  $context(v_i, 3)$  and  $context(w_j, 3)$  vectors and the cosine similarity between.

Classification experiments on SemEval-2021 ar-ar datasets were conducted using the following

two CBOV Word2Vec models and two corresponding Lemma2Vec models: (i) Wiki-CBOV, a pre-trained Word2Vec model from the set of AraVec models (Soliman et al., 2017), (ii) our CBOV Word2Vec model that we trained ourselves, (iii) Lemma2Vec model that we constructed based on the Wiki-CBOV model, and (iv) Lemma2Vec that we constructed based on our CBOV Word2Vec model. Based on these four models, four experiments were conducted to tune the following parameters: context size (*context\_size*), *threshold*, pooling operation (*pooling*) and removing of functional words (*stop\_words*).

### 4 Corpus and trained Word Embeddings

Two CBOV Word2Vec models were used in our experiments. The Wiki-CBOV (Soliman et al., 2017), which consists of 234,173 vocabulary size, and another model we trained our self which consists of 334,161 vocabulary size. The Wiki-CBOV model was learnt from a corpus of Arabic Wikipedia articles of about 78 million words, the principal hyperparameters are: 5 for minimum word count and 5 for window size.

Our CBOV Word2Vec model was trained on Modern Standard Arabic corpora, such as (El-Khair, 2017; Abbas and Smaili, 2005; Abdelali et al., 2014) of about 3 billion words; it was fit using 300-dimensional word vectors, 100 the minimum count of words, training epochs of 5 and window size of 5.

Before training the Word2Vec model, several normalization and preprocessing steps were performed. First, all diacritics, punctuations, Madda character, digits (Hindi and Arabic), Latin characters (including accented letters) were removed. Second, some special Arabic letters are unified. Third, sequences of repeated characters with length larger than 2 were reduced to one character; repeated spaces were also replaced by one space. Fourth, different forms of Alifs (ا | آ | إ) are replaced with (ا). Spaces followed by a period character and new lines were considered to be end of sentence marks. The split method in Python is used in tokenization. The vocabulary size of the resulted model is 334,161.

### 5 Constructing the Lemma2Vec models

Two Lemma2Vec models were produced, based on both: the Wiki-CBOV Word2Vec model, and our CBOV Word2Vec model. Each vocabulary in

each of the Word2Vec models was lemmatized first. Then a vector for each lemma (*i.e.*, Lemma2Vec) is calculated as following: first all word forms belonging to this lemma are fetched, then their Word2Vec vectors are combined through a *mean* pooling operation. The lemmatization process was performed using in-house tools and lexicographic databases <sup>1</sup> belonging to Birzeit University (Jarrar, 2021; Jarrar and Amayreh, 2019; Jarrar et al., 2019). In case of a word cannot be lemmatized due to misspelling, incorrect tokenization or in case of foreign word (not included in our database), then the corresponding Lemma2Vec is considered to be its Word2Vec vector.

Table 1 summarizes the lemmatization results that we performed on both, the Wiki-CBOW model and our CBOW Word2Vec model. The lemmatized words of SemEval-2021 all *ar-ar* dataset, as well as the Word2Vec and Lemma2Vec of *ar-ar* datasets words’s vectors used in this paper are available on-line <sup>2</sup>.

	Wiki-CBOW	Our Word2Vec
	78M words min_count 5	3B words min_count 100
Unique word forms	234,173	334,161
Unique lemmas	100,040	54,788
Words not lemmatized	22,054	28,098

Table 1: Lemmatization results for both models.

## 6 Experiments Results and Discussion

Given our Arabic WiC disambiguation method described in Section 3, and given the SemEval multilingual *dev.ar-ar* dataset provided by SemEval-2021 (Martelli et al., 2021), four classification experiments were conducted using the cosine similarity and based on the two Word2Vec models and the two Lemma2Vec models. The objective is to tune the following parameters for each model: *context\_size* (ranging from 1 to 10), *threshold* (we determined empirically the range from 0.55 to 0.85 with 0.1 step size), *pooling* (*min*, *max*, *mean* and *std*), and *stop\_words* (*yes*, *no*). Then the values of parameters corresponding to the high F1-scores for TRUE (T) and FALSE (F) classes are selected in order to classify sentence pairs in the *test.ar-ar* dataset. For each model we did

<sup>1</sup><https://ontology.birzeit.edu>

<sup>2</sup>[https://ontology.birzeit.edu/semEval2021\\_data.zip](https://ontology.birzeit.edu/semEval2021_data.zip)

	Exp1	Exp2	Exp3	Exp4
<b>Model</b>	Word2Vec Wiki-CBOW	Lemma2Vec Wiki-CBOW	Word2Vec our model	Lemma2Vec our model
<i>context_size</i>	4	1	4	1
<i>pooling</i>	<i>min</i>	<i>min</i>	<i>min</i>	<i>mean</i>
<i>threshold</i>	0.66	0.56	0.83	0.58
<i>stop_words</i>	<i>yes</i>	<i>yes</i>	<i>no</i>	<i>yes</i>
<b>Dataset</b>	<i>dev.ar-ar</i>			
<b>Tag</b>	T	F	T	F
Precision	52	52	57	58
Recall	54	51	61	53
F1-score	53	52	59	56
<b>Dataset</b>	<i>test.ar-ar</i>			
Accuracy	57	59	59	<b>60</b>

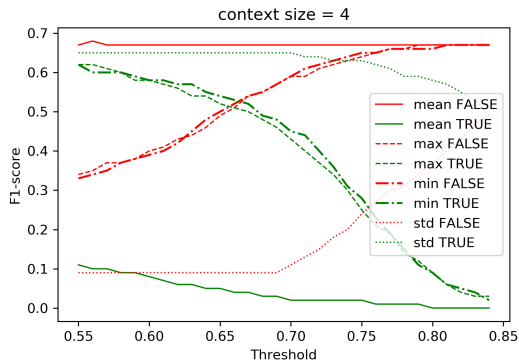
Table 2: Best F1-score, precision and recall values of the four experiments on *dev.ar-ar* dataset with the values of tuned parameters. Below are accuracies on *test.ar-ar* dataset.

the following to find the high F1-scores for T and F: For each *context\_size* (between 1 and 10) and for each value of the *stop\_words* (*yes* or *no*) we plotted 8 line plots (4 for T and 4 for F) for each of the four pooling operations (*mean*, *max*, *min* and *std*) and for *threshold* ranging from 0.55 to 0.85 (*i.e.*, 20 plots for each model, resulting 80 plots).

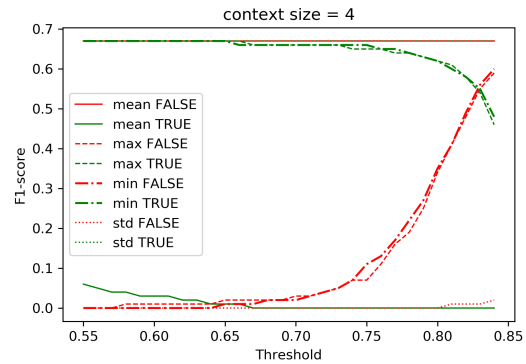
Figures 3a, 3b, 3c and 3d show the best 4 F1-scores line plots for each of the four models, and Table 2 shows the effective F1-scores values for T and F classes as well as precision and recall values (best results marked in bold). The values of parameters corresponding to the best result were then used in classifying the *test.ar-ar* dataset. The accuracies are reported in Table 2 as well.

As shown in Figure 3, the Lemma2Vec models have the tendency to perform better with shorter context sizes compared with the Word2Vec models. A possible reason may be that, in case of Lemma2Vec, the narrow meaning of words is affected due to the increase number of words involved in Lemma2Vec vector calculation. The impact of Lemma2Vec on the narrow meaning of words is discussed in the next subsection.

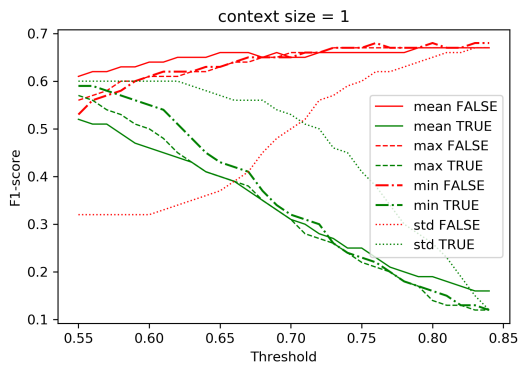
The results with *yes* for *stop\_words* are slightly better but not significant. Additionally, the *min* pooling was generally the best operation to combine the context vectors, and the results of both *min* and *max* pooling were close to each other.



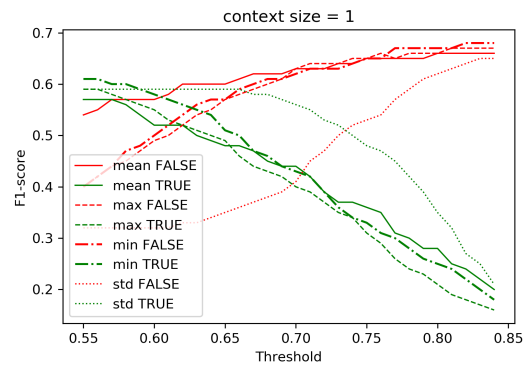
(a) **Wiki-CBOW Word2Vec model.**  
*context\_size = 4 - pooling = min*  
*threshold = 0.66 - stop\_words = yes*



(b) **our Word2Vec model.**  
*context\_size = 4 - pooling = min*  
*threshold = 0.83 - stop\_words = no*



(c) **Wiki-CBOW Lemma2Vec model.**  
*context\_size = 1 - pooling = min*  
*threshold = 0.56 - stop\_words = yes*



(d) **our Lemma2Vec model.**  
*context\_size = 1 - pooling = mean*  
*threshold = 0.58 - stop\_words = yes*

Figure 3: The best four F1-scores markers plots for each of the four models. The values of parameters are under each plot.

## 6.1 Lemma2Vec-Word2Vec Error Analyses

This subsection discusses the performance of using lemma-based vs. word-based models in the WiC disambiguation task, which we summarize in Table 3 and Table 4.

	TRUE	FALSE	Total
Correct L2V - Correct W2V	225	145	370
<b>Correct L2V - Wrong W2V</b>	118	98	216
<b>Wrong L2V - Correct W2V</b>	66	116	182
Wrong L2V - Wrong W2V	91	141	232
Total	500	500	1000

Table 3: Wiki-CBOW Lemma2Vec vs. Word2Vec

Table 3 presents the results of experiments 1 and 2 (using Word2Vec and Lemma2Vec of Wiki-CBOW) whereas Table 4 presents the results of experiments 3 and 4 (using Word2Vec and Lemma2Vec of our CBOW model). In each table, we compare between cases that were correctly or wrongly classified by both models. For example, the second row in Table 3 shows that 216 sentence pairs (118 TRUE class + 98 FALSE

	TRUE	FALSE	Total
Correct L2V - Correct W2V	124	241	365
<b>Correct L2V - Wrong W2V</b>	188	45	233
<b>Wrong L2V - Correct W2V</b>	58	178	236
Wrong L2V - Wrong W2V	130	36	166
Total	500	500	1000

Table 4: Our Lemma2Vec vs. our Word2Vec

class) were correctly classified using the Wiki-CBOW’s Lemma2Vec model but wrongly classified using the Word2Vec. Similarly, 182 sentence pairs in the third row were correctly classified using the Word2Vec but wrongly classified using the Lemma2Vec.

As shown in both tables’ second and third rows, the Lemma2Vec did not significantly improve the overall results; but notably, the Lemma2Vec shows a significant improvement over Word2Vec for TRUE class whereas Word2Vec is better for FALSE class.

This conclusion is valid for all models, whatever are the corpora content, size and *min\_count*

test.ar-ar.342	(Correct with Lemma2Vec – Wrong with Word2Vec)
والنظام الحالي للدورات يسمح بمراعاة مختلف مواقف الوفود، ويتمتع بصوتة معينة..... sentence1: ..... عن الميزانية وسرعة برامج المنظمة <b>ومرونتها</b> من محدودية الموارد المتاحة في الميزانية العادية..... sentence2: ..... Class: TRUE	
test.ar-ar.994	(Wrong with Lemma2Vec – Correct with Word2Vec)
يتميز بقلة أو عدم وجود تخييلات جنسية والرغبة في ممارسة الجنس لفترة من الزمن. sentence1: ..... الذي يميز الرواية عن باقي الأجناس الأدبية الثرية الأخرى، وإنما توجد مقومات فنية أخرى... sentence2: ..... Class: FALSE	

Figure 4: Example of errors.

hyperparameter.

To understand the gain and loss by the lemma-based models, we manually analyzed most cases. Figure 4 illustrates such cases. The first sentence pair in Figure 4 was correctly classified by the Lemma2Vec (in **Exp4**) and wrongly by the Word2Vec (in **Exp3**). This illustrates that the lemma vector as a generalized model for its inflections (*i.e.*, a mean of word forms' vectors) was better in deciding that both contexts are similar and that the two word forms have the same meaning. However, the second example in Figure 4 illustrates the other way. The Lemma2Vec was too general, and the Word2Vec was specific enough, to decide that the two word forms, in the two contexts, are different. The word from al-ğins (الجنس) could mean both *genus* and *sex*; however the other word form al-ʾağnās (الأجناس), is semantically distinctive by its own morphology - as it can only be plural of *genus*, and cannot be plural of *sex*.

To conclude, although Lemma2Vec outperforms Word2Vec in some cases (mostly in the TRUE sentence pairs class), it underperforms Word2Vec in others cases (mostly in the FALSE sentence pairs class). Since the distribution of TRUE and FALSE is equal in the datasets, the overall performance of both models is close to each other. Nevertheless, in case of an application scenario where a large proportion of sentence pairs is expected to be TRUE, we recommend the use of Lemma2Vec, otherwise the Word2Vec.

## 7 Conclusions and Further Work

We presented a set of experiments to evaluate the performance of using Word2Vec and Lemma2Vec models in Arabic WiC disambiguation, without using external resources or any context/sense embedding model. Different models were constructed based on two different corpora, and different types of parameters were tuned. The final results demonstrated that Lemma2Vec models are slightly better than Word2Vec models for Arabic WiC disambiguation. More specifically, we found that

Lemma2Vec outperforms Word2Vec for TRUE sentence pairs, but underperforms it for FALSE sentence pairs.

We plan to extend our work to use our Lemma2Vec model to build a multi-prototype embeddings using the large lexicographic database available at Birzeit University. We plan also to fine tune the recently released Arabic BERT models, such as (Safaya et al., 2020; Antoun et al., 2020; Abdelali et al., 2021; Inoue et al., 2021), using the same database.

## Acknowledgments

We would like to thank the shared task organizers and the reviewers for their valuable comments and efforts towards improving our manuscript. We would like to also thank Taymaa Hammouda for her technical support.

## References

- Mourad Abbas and Kamel Smaili. 2005. Comparison of topic identification methods for arabic language. In *Proceedings of International Conference on Recent Advances in Natural Language Processing, RANLP*, pages 14–17.
- Ahmed Abdelali, Francisco Guzman, Hassan Sajjad, and Stephan Vogel. 2014. The amara corpus: Building parallel language resources for the educational domain. In *LREC*, volume 14, pages 1044–1054.
- Ahmed Abdelali, Sabit Hassan, Hamdy Mubarak, Kareem Darwish, and Younes Samih. 2021. [Pre-training bert on arabic tweets: Practical considerations](#).
- Mohammad Khaled A Al-Maghasbeh and MP Bin Hamzah. 2015. Extract the semantic meaning of prepositions at arabic texts: an exploratory study. *Int J Comput Trends Technol*, 30(3):116–120.
- Ali Alkhatlan, Jugal Kalita, and Ahmed Alhaddad. 2018. Word sense disambiguation for arabic exploiting arabic wordnet and word embedding. *Procedia computer science*, 142:50–60.
- Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. Arabert: Transformer-based model for arabic language understanding. In *LREC 2020 Workshop Language Resources and Evaluation Conference*, page 9.
- Nadia Bouhriz, Faouzia Benabbou, and EH Ben Lahmar. 2016. Word sense disambiguation approach for arabic text. *International Journal of Advanced Computer Science and Applications*, 7(4):381–385.

- Ibrahim Bounhas, Raja Ayed, Bilel Elayeb, Fabrice Evrard, and Narjes Bellamine Ben Saoud. 2015. Experimenting a discriminative possibilistic classifier with reweighting model for arabic morphological disambiguation. *Computer Speech & Language*, 33(1):67–87.
- KZ Bousmaha, S Charef Abdoun, L Hadrich Belguith, and MK Rahmouni. 2013. Une approche de désambiguïsation morpho.lexicale évaluée sur l’analyseur morphologique alkhalil. *Revue RIST—Vol*, 20(2):33.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Ibrahim Abu El-Khair. 2017. Abu el-khair corpus: A modern standard arabic corpus. *International Journal of Recent Trends in Engineering & Research (IJRTER)*, 03(1):95–100.
- Bilel Elayeb. 2019. Arabic word sense disambiguation: a review. *Artificial Intelligence Review*, 52(4):2475–2532.
- Jeremy Howard and Sebastian Ruder. 2018. [Fine-tuned language models for text classification](#). *CoRR*, abs/1801.06146.
- Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. The interplay of variant, size, and task type in Arabic pre-trained language models. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, Kyiv, Ukraine (Online). Association for Computational Linguistics.
- Mustafa Jarrar. 2021. [The arabic ontology - an arabic wordnet with ontologically clean content](#). *Applied Ontology Journal*, 16(1):1–26.
- Mustafa Jarrar and Hamzeh Amayreh. 2019. [An arabic-multilingual database with a lexicographic search engine](#). In *International Conference on Applications of Natural Language to Information Systems*, volume 11608 of *LNCS*, pages 234–246. Springer.
- Mustafa Jarrar, Hamzeh Amayreh, and John P McCrae. 2019. [Representing arabic lexicons in lemon - a preliminary study](#). In *The 2nd Conference on Language, Data and Knowledge (LDK 2019)*, volume 2402, pages 29–33. CEUR.
- Mustafa Jarrar, Fadi Zaraket, Rami Asia, and Hamzeh Amayreh. 2018. [Diacritic-based matching of arabic words](#). *ACM Transactions on Asian and Low-Resource Language Information Processing (TAL-LIP)*, 18(2):10:1–10:21.
- Dan Klein, Kristina Toutanova, H Tolga Ilhan, Sepandar D Kamvar, and Christopher D Manning. 2002. Combining heterogeneous classifiers for word sense disambiguation. In *Proceedings of the ACL-02 workshop on Word sense disambiguation: recent successes and future directions*, pages 74–80.
- Rim Laatar, Chafik Aloulou, and Lamia Hadrich Belguith. 2017. Word sense disambiguation of arabic language with word embeddings as part of the creation of a historical dictionary. In *LPKM*.
- Yoong Keok Lee and Hwee Tou Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 41–48.
- Massimiliano Mancini, Jose Camacho-Collados, Ignacio Iacobacci, and Roberto Navigli. 2016. Embedding words and senses together via joint knowledge-enhanced training. *arXiv preprint arXiv:1612.02703*.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Larousi Merhbene, Anis Zouaghi, and Mounir Zrigui. 2014. Approche basée sur les arbres sémantiques pour la désambiguïsation lexicale de la langue arabe en utilisant une procédure de vote. In *Proceedings of the 21st conference on natural language processing (TALN 2014)*, Marseille, France, pages 281–290.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. 2017. Making sense of word embeddings. *arXiv preprint arXiv:1708.03390*.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *CoRR*, abs/1910.10683.
- Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. [KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059.

Barcelona (online). International Committee for Computational Linguistics.

Rana Aref Salama, Abdou Youssef, and Aly Fahmy. 2018. Morphological word embedding for arabic. *Procedia computer science*, 142:83–93.

Pamela Shapiro and Kevin Duh. 2018. Morphological word embeddings for arabic neural machine translation in low-resource settings. In *Proceedings of the Second Workshop on Subword/Character Level Models*, pages 1–11.

Abu Bakr Soliman, Kareem Eissa, and Samhaa R El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265.

Nadia Soudani, Ibrahim Bounhas, Bilel ElAyeb, and Yahya Slimani. 2014. Generic normalization approach of arabic dictionaries for arabic word sense disambiguation. *Proceedings of Cinquième Journées Francophones sur les Ontologies (JFO), Hammamet, Tunisia*, pages 309–315.

# GlossReader at SemEval-2021 Task 2: Reading Definitions Improves Contextualized Word Embeddings

Maxim Rachinskiy<sup>◇</sup> and Nikolay Arefyev<sup>△▽◇</sup>

<sup>◇</sup>HSE University, Moscow, Russia

<sup>△</sup>Samsung Research Center Russia, Moscow, Russia

<sup>▽</sup>Lomonosov Moscow State University, Moscow, Russia

myurachinskiy@edu.hse.ru narefjev@cs.msu.ru

## Abstract

Consulting a dictionary or a glossary is a familiar way for many humans to figure out what does a word in a particular context mean. We hypothesize that a system that can select a proper definition for a particular word occurrence can also naturally solve tasks related to word senses. To verify this hypothesis we developed a solution for the Multilingual and Cross-lingual Word-in-Context (MCL-WiC) task, that does not use any of the shared task data or other WiC data for training. Instead, it is trained to embed word definitions from English WordNet and word occurrences in English texts into the same vector space following an approach previously proposed by Blevins and Zettlemoyer (2020) for Word Sense Disambiguation (WSD). To estimate the similarity in meaning of two word occurrences, we compared different metrics in this shared vector space and found that L1-distance between normalized contextualized word embeddings outperforms traditionally employed cosine similarity and several other metrics. To solve the task for languages other than English, we rely on zero-shot cross-lingual transfer capabilities of the multilingual XLM-R masked language model. Despite not using MCL-WiC training data, in the shared task our approach achieves an accuracy of 89.5% on the English test set, which is only 4% less than the best system. In the multilingual subtask zero-shot cross-lingual transfer shows competitive results, that are within 2% from the best systems for Russian, French, and Arabic. In the cross-lingual subtask are within 2-4% from the best systems.

## 1 Introduction

SemEval-2021 Task 2 is a multilingual and cross-lingual word-in-context disambiguation task (MCL-WiC) for five different languages (Martelli et al.,

2021).<sup>1</sup> Each example in the multilingual subtask consists of two sentences in English, Russian, French, Arabic, or Chinese language containing occurrences of the same target word. In the cross-lingual subtask each example consists of two sentences in different languages containing occurrences of two different target words. The participants were asked to detect whether those occurrences corresponded to the same or different meanings. These tasks are formalized as binary classification tasks. The datasets contain the same number of examples for each class. Accuracy is utilized as the main performance metric.

Recent SOTA approaches to the WiC task mainly include fine-tuning large universally pre-trained masked language models (Raffel et al., 2020; Liu et al., 2019) on labeled WiC datasets. Instead, we decided to train a system that selects the most appropriate definition for each word occurrence following an approach proposed by Blevins and Zettlemoyer (2020) for Word Sense Disambiguation (WSD) and experimented with different ways of adapting such system to the MCL-WiC task. During the evaluation period, we experimented with distances between probability distributions over word definitions but did not manage to achieve good results. But through the post-evaluation period, we switched to distances between the contextualized word embeddings of our gloss-informed language model and improved our results significantly achieving comparable results with the 2nd best system for French and 6th best system for Arabic in the multilingual subtask.

Our main interest was whether the word-in-context systems can benefit from using gloss information, provided for each possible sense of the word.

<sup>1</sup><https://competitions.codalab.org/competitions/27054>



## 2 Background

Here we summarize prior work linking word occurrences and word definitions. One of the first approaches in this field (Lesk, 1986) calculated the lexical overlap between the context of a particular word occurrence and all possible definitions of this word. This approach did not take into account word synonymy or other lexical relations. The following work tried to combine state-of-the-art language models with glosses from some dictionaries.

One of such methods has been proposed by Kumar et al. (2019). Their EWISE system used a pre-training procedure for a gloss encoder, that learned knowledge graph embeddings from WordNet (Miller, 1995). After this pre-training, the authors froze the gloss encoder and started to train a context encoder with labeled WSD data. While the method of Kumar et al. (2019) requires relational information from a knowledge graph, the method proposed by Huang et al. (2019) relies fully on gloss information. The developed system jointly encodes the context with all possible glosses of the target word. The authors used a pre-trained BERT (Devlin et al., 2019) model as initialization for their encoder.

A similar approach has been proposed by Blevins and Zettlemoyer (2020), who trained two separate Transformer-based encoders for word occurrences (Context encoder) and word definitions (Gloss encoder), both initialized with BERT weights (Devlin et al., 2019). To represent a word occurrence, the outputs of the Context encoder for all of its subwords were averaged. To represent a definition, the output of the Gloss encoder from [CLS] token was taken. Finally, for a word occurrence and all of its definitions, the dot products between those outputs were calculated and the softmax function was applied to them, resulting in a probability distribution over possible word senses. The whole model was trained using cross-entropy loss to select the correct word sense on WSD data.

## 3 System overview

In order to learn sense-dependent representations of words, we pre-train our system on the Word Sense Disambiguation task. Following the BEM model (Blevins and Zettlemoyer, 2020), our system consists of two separate encoders: Context Encoder and Gloss Encoder.

**Context encoder ( $T_c$ )** takes a sentence  $c = c_0, \dots, c_{i-1}, w_c, c_{i+1}, \dots, c_n$  containing a target word  $w_c$  to be disambiguated, where  $w_c$  is the  $i^{th}$  word in the sentence. The encoder then produces the target word representation:

$$r_{w_c} = T_c(c)[i]$$

For target words that are tokenized into multiple subword units, we average representations of these subwords.

**Gloss encoder ( $T_g$ )** takes as input a gloss  $g_s$  that defines a word sense  $s$  and encodes it as:

$$r_s = T_g(g_s)[0]$$

Taking the output from the first input token, which should be [CLS] for BERT or <s> for XLM-R.

We can score each of the possible senses  $s \in S_w$ , for a target word  $w_c$  by taking the dot product of  $r_{w_c}$  against every  $r_s$  for  $s \in S_w$ :

$$\phi(w_c, s) = r_{w_c}^T r_s$$

Both encoders were initialized with BERT or XLM-R weights. Then the whole system was pre-trained on English WSD data (Miller et al., 1994) with cross-entropy loss. We denote this pre-training procedure as Gloss Language Modeling (GLM) and compare it with pure Masked Language Modeling (MLM) pre-training. In both cases, the models were not fine-tuned on any MCL-WiC data.

### 3.1 Adaptation to the MCL-WiC task

As EWISE (Kumar et al., 2019) and BEM (Blevins and Zettlemoyer, 2020) systems work only with English data, we extend the proposed approach to the multilingual setting by replacing BERT with XLM-R model (Conneau et al., 2019). In the result section, we discuss how this affects the resulting performance.

Here we present two approaches to the final MCL-WiC task, one using distributions over possible word definitions and another exploiting different similarity measures between contextualized target word embeddings from the Context Encoder. The latter is also applicable to the contextualized word embeddings obtained from MLM pre-trained XLM-R, which we consider as a baseline.

#### 3.1.1 Probability distribution over glosses

Here we exploit probability distributions  $P(\text{sense}|w_{c_1})$  and  $P(\text{sense}|w_{c_2})$  produced by

Sentence	gloss #1	gloss #2	gloss #3
dev.en-en.1, Meanings are the same			
No clause in a contract shall be interpreted as evading the responsibility of <b>superiors</b> under international law	one of greater rank or station or quality (0.82)	the head of a religious community (0.12)	a combatant who is able to defeat rivals (0.06)
In Senegal too, the customs officer and his <b>superiors</b> receive a premium in case of detecting and preventing smuggling	one of greater rank or station or quality (0.58)	the head of a religious community (0.3)	a combatant who is able to defeat rivals (0.09)
dev.en-en.16, Meanings are different			
During the fight both of them <b>tripped</b> , the author falling on the victim and stabbing him with the knife by accident	miss a step and fall or nearly fall (0.998)	cause to stumble (0.0009)	put in motion or move to act (0.0004)
The father of the child also cannot take the child to <b>trip</b> during the fostering duration, without permission of fosterer	make a trip for pleasure (0.9)	miss a step and fall or nearly fall (0.06)	get high, stoned, or drugged (0.02)

Table 1: Examples from the MCL-WiC development set with 3 most probable glosses of the target word predicted by our system. Word in **bold** is the target word. The rounded probabilities for each of the meanings are given in parentheses.

our WSD system for words  $w_{c_1}$  and  $w_{c_2}$  in their contexts  $c_1$  and  $c_2$ .  $w_{c_1}$  and  $w_{c_2}$  have the same lemma and consequently have the same set of possible meanings  $S_w$  from the vocabulary.

**Gloss match prob:** The probability that two word occurrences,  $w_{c_1}$  and  $w_{c_2}$ , have the same meaning (positive class) is calculated as:

$$P(1|w_{c_1}, w_{c_2}) = \sum_{s_i \in S_w} P(s_i|w_{c_1}) \cdot P(s_i|w_{c_2})$$

**Gloss JSD:** As an alternative measure of word similarity in context, we compute Jensen–Shannon divergence between two distributions  $P(\text{sense}|w_{c_1})$  and  $P(\text{sense}|w_{c_2})$ .

Because these methods rely on gloss information, we need a vocabulary to find them. As for English, we can easily use WordNet (Miller, 1995), it becomes problematic to find definitions for other languages. To counteract this obstacle during the competition we used the following procedure. First, we translated all samples from other languages to English via machine translation<sup>2</sup>. Then we generated all possible translations of the target word with Yandex Translation API<sup>3</sup> and word2word library<sup>4</sup>. Finally, we tried to find one of the possible target word translations in the translated sentence. If there was no match for both sentences, we predicted True, if a match was only for one, we predicted False. In case of more than one match in the translated sentence, we took the first one. In the end, we had two

<sup>2</sup>[https://huggingface.co/transformers/model\\_doc/marian.html](https://huggingface.co/transformers/model_doc/marian.html)

<sup>3</sup><https://yandex.com/dev/dictionary/>

<sup>4</sup><https://github.com/kakaobrain/word2word>

occurrences of possibly different translations of the target word into English and could use previous metrics. But unlike the original English method where we need to disambiguate between meanings of the same word, here we build the distribution over all possible glosses of all possible translations of the target word.

### 3.1.2 Similarity between contextualized word embeddings

In this subsection, we propose methods that fully rely on the Context encoder and thus do not require any additional vocabulary or glosses. We achieve such generalization by using only outputs from the trained Context encoder.

**Cosine:** Cosine similarity between outputs of the encoder.

**Euclidian+norm:** Euclidian distance between L2 normalized outputs of the encoder.

**Manhattan+norm:** Manhattan distance between L1 normalized outputs of the encoder.

## 3.2 Threshold selection

As MCL-WiC is a binary classification problem, we need to transform our continuous similarity measures into binary predictions. We select the best threshold with grid search on one of the following datasets.

1. (**xx dev**) The threshold is selected on the development set of the target language.
2. (**en dev**) The threshold for all languages is selected on the English development set.

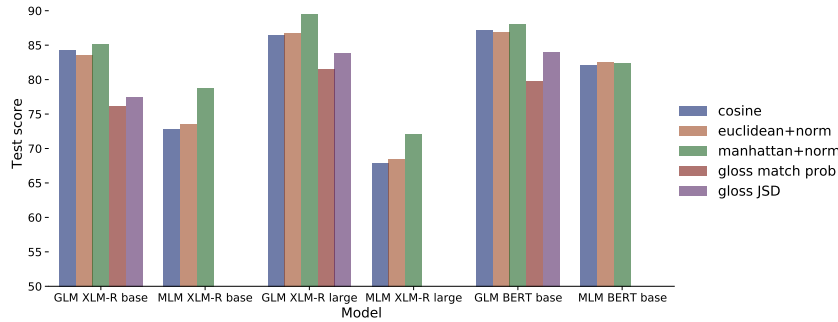


Figure 1: Comparison of pre-training methods and similarity measures on the English MCL-WiC test set.

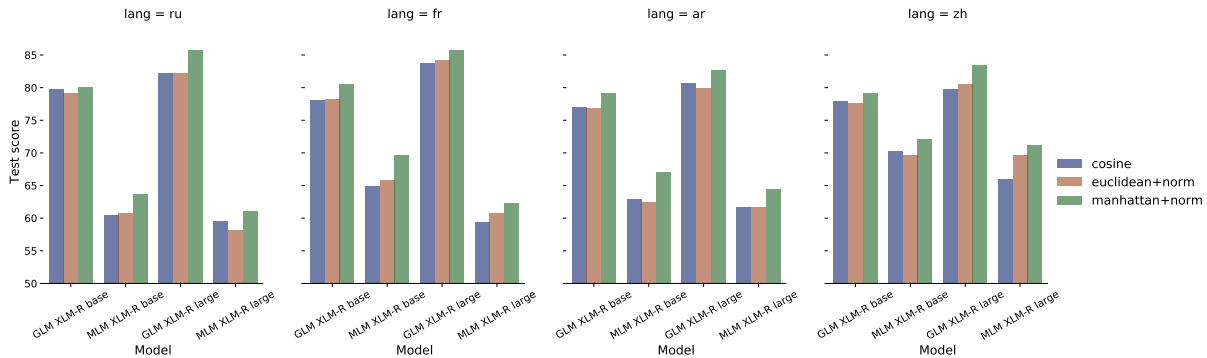


Figure 2: Comparison of pre-training methods and similarity measures on MCL-WiC test sets.

3. (**semcor**) Instead of utilizing MCL-WiC development data, we tried estimating the optimal threshold on the training SemCor dataset. Based on sense annotations, we constructed 30K sentence pairs in WiC format. This resulted in a nearly-balanced WiC dataset, which was used for grid search.
4. (**cl trials**) The threshold for the cross-lingual subtask is selected on the concatenated cross-lingual MCL-WiC trial sets.

## 4 Experimental setup

Besides choosing the best final predictor for MCL-WiC, we also experimented on encoder initialization. In the result section, we compare the performance of the models, initialized with BERT base (Devlin et al., 2019), XLM-R base, and XLM-R large (Conneau et al., 2019). We trained our models on the English SemCor dataset (Miller et al., 1994) with glosses from the WordNet 3.0 (Miller, 1995). Systems based on the XLM-R base and XLM-R large (Conneau et al., 2019) were trained 20 and 10 epochs respectively. Following standard practices, we used SemEval-2007 (Pradhan et al., 2007) as our development set for early stopping. For the

system with BERT-base, we used the originally provided checkpoint by Blevis and Zettlemoyer (2020).

## 5 Results

### 5.1 Similarity measures

The results of the experiments with different similarity measures for English and non-English languages are given in Figure 1 and Figure 2 respectively. Figure 1 shows that approaches based on GLM context outputs strongly outperform methods based on distributions over senses from the vocabulary.

Figures 1, 2 also provide an observation, that almost for any language and model Manhattan+norm distance shows the best results. The only exception is the MLM BERT base model, where Euclidean+norm performs slightly better.

### 5.2 GLM vs MLM

Figure 4 shows the gap between GLM and pure MLM pre-training. Experiments show that models trained with GLM procedure strongly outperform their MLM counterparts in every language and with any base model.

Model	en	ru	fr	ar	zh
our post-evaluation results					
GLM XLM-R base - Manhattan+norm	85.1	80.1	80.5	79.2	79.1
MLM XLM-R base - Manhattan+norm	78.8	63.6	69.6	67.1	72.1
GLM XLM-R large - Manhattan+norm (en dev)	<b>89.5</b>	<b>85.7</b>	<b>86.5</b>	<b>84.2</b>	<b>83.5</b>
GLM XLM-R large - Manhattan+norm (semcor)	87.8	82.7	84.2	80.9	80.8
GLM XLM-R large - Manhattan+norm	<b>89.5</b>	<b>85.7</b>	85.7	82.6	<b>83.5</b>
MLM XLM-R large - Manhattan+norm	72.1	61.1	62.3	64.5	71.1
GLM BERT base - Manhattan+norm	88	-	-	-	-
MLM BERT base - Euclidean+norm	82.5	-	-	-	-
our submissions					
GLM BERT base - Gloss JSD	86.4	-	-	-	-
GLM BERT base (MT) - Gloss JSD	86.4	68.3	69.2	50.1	64.1
best submissions					
Best for each lang.	<b>93.3</b>	<b>87.4</b>	<b>87.5</b>	<b>84.8</b>	<b>91</b>

Table 2: Best test score for each of the proposed systems. (*MT*) states for the Machine Translation technique, described in Section 3.1. The threshold for binary classification was calculated either on the English dev set (*en dev*), or on a part of SemCor dataset (*semcor*), or on the dev set, corresponding to the target language (all others). *Best for each lang.* stands for the best results of the competition for each of the languages.

Surprisingly, the XLM-R base model outperforms the large one when pre-trained with MLM objective only. However, after GLM pre-training the large model performs significantly better than the base model. We suspect that this is due to the strong grammatical bias of contextualized word embeddings after MLM pre-training also observed by Laicher et al. (2021). It is easier to correctly predict the grammatical form of a masked word in a particular context than the exact lemma of that word. Thus, the model is much more confident in the grammatical form resulting in distant embeddings for the same word in the same sense but in different grammatical forms. Since the large model shall better optimize the MLM objective, its embeddings likely contain stronger grammatical component hiding the sense component relevant for the MCL-WiC task. Since word definitions correspond to word senses and not word forms, the GLM objective helps eliminating the irrelevant grammatical component from contextualized embeddings.

### 5.3 Correlation between WSD and MCL-WiC performance

We also suspected that during pre-training on the English WSD data, GLM models can overfit to English texts and partially lose their cross-lingual transferability. Thus, the best epoch checkpoint based on WSD development set may not be the best in terms of MCL-WiC. In Figure 3 we show how MCL-WiC accuracy for each language and WSD F1 score for English change during training. Multilingual WiC performance for epoch 0 stands for the MLM model without any GLM training.

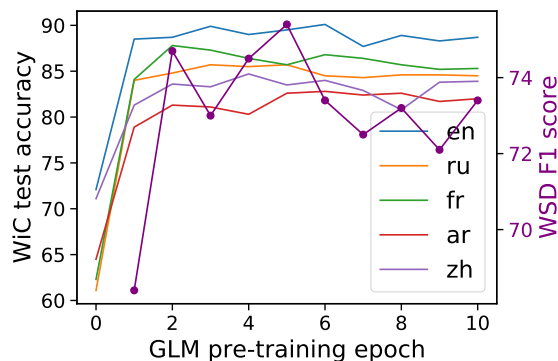


Figure 3: Test MCL-WiC score and SemEval07 dev F1 score for the XLM-R large model depending on the epoch.

The results show that choosing the best checkpoint by WSD F1 score, we get nearly optimal results on MCL-WiC for each language, except for French.

### 5.4 Interpreting system predictions

As our system embeds word definitions from WordNet (Miller, 1995) and word occurrences into the same vector space, we can search for the nearest definitions for each occurrence of the target word. In Table 1 we show some examples from the development set with top3 senses (glosses) predicted by our system for each occurrence. We used GLM XLM-R large as a backbone for this purpose.

### 5.5 Overall multilingual results

Table 2 shows overall results on the competition’s test set. The submission *GLM BERT base - Gloss JSD* sent during the competition employed English BERT base (Devlin et al., 2019) backbone in the

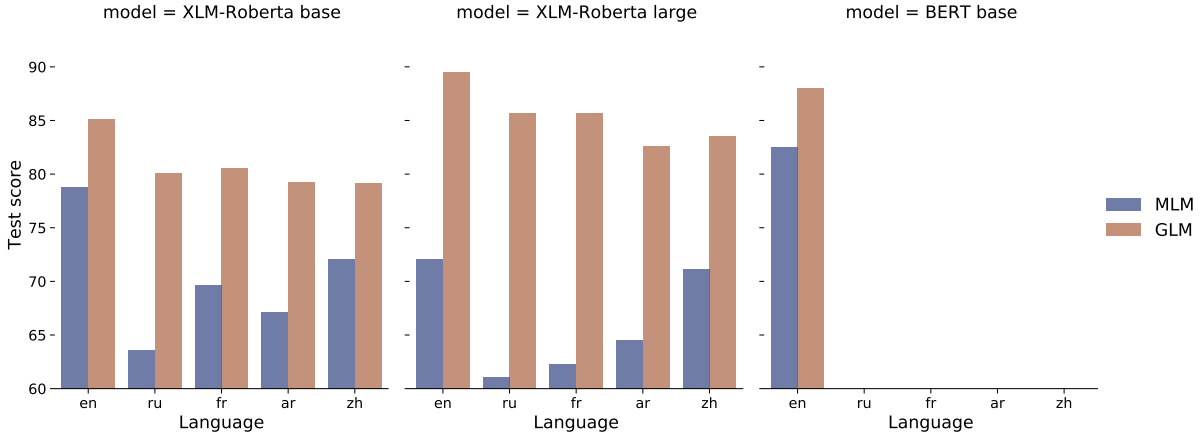


Figure 4: Test score for the best GLM and MLM models for each language.

WSD model, which was applicable to English data only. The submission *GLM BERT base (MT) - Gloss JSD* exploits the same basic idea but extends the first approach to other languages with the translation technique, described in Section 3.1. For both submissions, due to a mistake, we used the GLM model trained for only one epoch on SemCor (Miller et al., 1994).

As we can see from the table 2, our best system on every language is GLM pre-trained XLM-R large model with Manhattan+norm distance and the threshold selected on the English MCL-WiC development set. This system shows rather strong results achieving the performance of the 2nd best system for French and the 6th best system for Arabic. Since this system does not use any non-English resources for training, we suspect that it will work for a variety of other languages on which XLM-R was initially pre-trained, though the performance may vary.

## 5.6 Cross-lingual results

Table 3 shows our post-evaluation results on the cross-lingual subtask of MCL-WiC. The threshold selected on the English dev set does not transfer to the cross-lingual test sets, unlike multilingual test sets. This is due to larger distances between contextualized embeddings returned by XLM-R for word occurrences in different languages. Selecting threshold on the concatenation of cross-lingual trial sets works much better. However, since there are only 32 cross-lingual examples, there is a wide interval of optimal thresholds giving the same accuracy on trial. Our implementation selected the smallest one, however, some larger thresholds resulted in a significant decrease in performance.

Model	en-ar	en-fr	en-ru	en-zh
our post-evaluation results				
(en dev)	77.6	81.5	81.8	78.9
(cl trials)	<b>85.2</b>	<b>85.5</b>	<b>87.2</b>	<b>89.2</b>
best submissions				
Best for each pair	<b>89.1</b>	<b>89.1</b>	<b>89.4</b>	<b>91.2</b>

Table 3: Post-evaluation test scores for the cross-lingual subtask. For each of our systems, we used GLM XLM-R large model and Manhattan+norm distance. *Best for each pair* stands for the best results of the competition for each pair of languages individually.

Thus, a larger cross-lingual development set is required for robust selection of the threshold.

## 6 Conclusion

In this paper, we presented Gloss Language Modeling (GLM) procedure as a pre-training strategy for MCL-WiC systems. We have shown that this procedure improves multilingual WiC performance on all languages for both XLM-R and BERT backbones.

Apart from that, we proposed an interpretable zero-shot multilingual WiC algorithm which does not require any labeled data for the multilingual WiC task except for the threshold selection, which can be performed using only English development data without loss of accuracy for other languages. We also found that L1-distance between normalized contextualized word embeddings outperforms traditionally employed cosine distance.

## Acknowledgments

This research was supported through computational resources of HPC facilities at NRU HSE.

## References

- Terra Blevins and Luke Zettlemoyer. 2020. [Moving down the long tail of word sense disambiguation with gloss informed bi-encoders](#). In *Proceedings of the 58th Association for Computational Linguistics*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. [GlossBERT: BERT for word sense disambiguation with gloss knowledge](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3507–3512, Hong Kong, China. Association for Computational Linguistics.
- Sawan Kumar, Sharmistha Jat, Karan Saxena, and Partha Talukdar. 2019. [Zero-shot word sense disambiguation using sense definition embeddings](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5670–5681, Florence, Italy. Association for Computational Linguistics.
- Severin Laicher, Sinan Kurtigit, Dominik Schlechtweg, Jonas Kuhn, and Sabine Schulte im Walde. 2021. [Explaining and improving BERT performance on lexical semantic change detection](#).
- Michael Lesk. 1986. [Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone](#). In *Proceedings of the 5th Annual International Conference on Systems Documentation, SIGDOC '86*, page 24–26, New York, NY, USA. Association for Computing Machinery.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#).
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. [SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation \(MCL-WiC\)](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- George A. Miller. 1995. [WordNet: A lexical database for English](#). *Commun. ACM*, 38(11):39–41.
- George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994. [Using a semantic concordance for sense identification](#). In *Proceedings of the Workshop on Human Language Technology, HLT '94*, page 240–243, USA. Association for Computational Linguistics.
- Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. [SemEval-2007 task-17: English lexical sample, SRL and all words](#). In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#).

# UAlberta at SemEval-2021 Task 2: Determining Sense Synonymy via Translations

**Bradley Hauer, Hongchang Bao, Arnob Mallik, Grzegorz Kondrak**  
Alberta Machine Intelligence Institute, Department of Computing Science  
University of Alberta, Edmonton, Canada  
{bmhauer, hongchan, amallik, gkondrak}@ualberta.ca

## Abstract

We describe the University of Alberta systems for the SemEval-2021 Multilingual and Cross-lingual Word-in-Context (MCL-WiC) disambiguation task. We explore the use of translation information for deciding whether two different tokens of the same word correspond to the same sense of the word. Our focus is on developing principled theoretical approaches which are grounded in linguistic phenomena, leading to more explainable models. We show that translations from multiple languages can be leveraged to improve the accuracy on the WiC task.

## 1 Introduction

This paper describes the University of Alberta systems for SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (Martelli et al., 2021). We focus on the monolingual (English) variant of the task, which is the same as the original WiC task (Pilehvar and Camacho-Collados, 2018). An instance of the WiC task consists of two sentences that share a *focus word* in common; the word may be inflected differently in each sentence (e.g. “they had searched his flat a few days before” and “the production of lithium from salt flats”) but will share the same lemma and part of speech. A WiC task system must decide, given such a pair of sentences, whether the *focus tokens* have the same meaning in both sentences. Systems are compared in terms of their accuracy, the percentage of test instances correctly identified as TRUE (same meaning) or FALSE (different meaning). The dataset includes training, development, and testing splits; as our methods are unsupervised, we do not use the training data.

The goal of this paper is an exploration of the use of translation information for the WiC task. The intuition underlying our work is that distinctions in meaning tend to be reflected in distinctions in

translation. We have previously presented methods leveraging translation information to improve word sense disambiguation (Luan et al., 2020), and most frequent sense detection (Hauer et al., 2019), and have demonstrated that word senses which share translations are, in general, semantically related (Hauer and Kondrak, 2020a). We have also presented theoretical formalizations of lexico-semantic phenomena which view synonymy and translation as two aspects of semantic equivalence (Hauer and Kondrak, 2020b). Our team additionally presented a method based on translation information (Hauer et al., 2020) for the SemEval-2020 Task 2 on Predicting Multilingual and Cross-Lingual Lexical Entailment (Glavaš et al., 2020). In this task, we investigate whether translation can be used to detect semantic equivalence in context, just as in the aforementioned prior task we investigated whether translation can be used to detect lexical entailment between word types. Our focus is on developing principled theoretical approaches which are grounded in linguistic phenomena, leading to more explainable models.

Our more complex methods depend upon a mapping between word senses and translations, as different senses of a word often translate differently. We obtain such a mapping from BabelNet (Navigli and Ponzetto, 2012), which combines information from Princeton WordNet (Fellbaum, 1998), multilingual lexical resources, and translations produced by MT models. WordNet is comprised of synonym sets, or *synsets*, which BabelNet enriches with translations. Each of the resulting multi-lingual synsets, or *multi-synsets*, contain lexicalizations of a single concept in various languages, allowing the translations of a given sense of a word to be identified. We treat BabelNet as an imperfect implementation of a universal multi-wordnet with the theoretical properties described by Hauer and Kondrak (2020b).

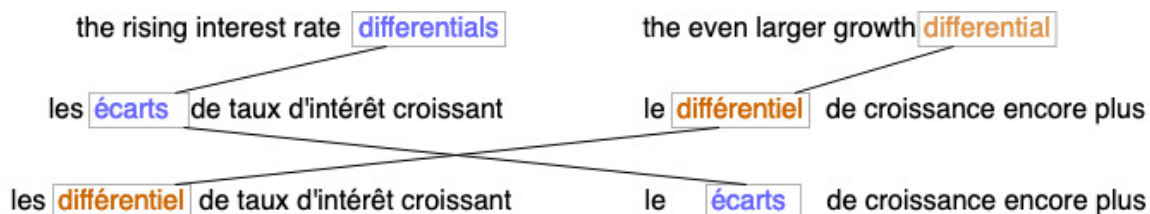


Figure 1: An example of the “translation criss-cross” described in Section 3.2.

Our results can be interpreted as a proof-of-concept for the use of contextual translations as indicators of semantic similarity. We show that the methods that we develop for the WiC task can leverage translations to improve over baselines, especially when multiple target languages are considered. While it is not our objective to compete with state-of-the-art supervised methods, we consider this to be a positive result, and a strong lead for future work on contextual semantic analysis.

This paper is structured as follows: Section 2 provides an overview of relevant prior literature. Section 3 discusses the theoretical model underlying our work. Section 4 outlines our methods. Section 5 describes our experiments and results.

## 2 Related Work

Methods for WiC task can be roughly divided into two paradigms: contextualized-embedding-based systems, and word sense disambiguation-based systems. Pilehvar and Camacho-Collados (2018) introduce the WiC dataset as a benchmark for evaluating context sensitive word representations. Soler et al. (2019) achieve improvements by combining similarity scores from different types of contextual word and sentence embeddings. Liu et al. (2020) propose a method to enhance contextual representations by leveraging other pre-trained contextual or static embeddings.

Another approach to WiC task is to employ a word sense disambiguation (WSD) system to tag the target words with senses from a pre-defined sense inventory and subsequently make a decision based on the predicted synsets of the target words. Loureiro and Jorge (2019b) use the LMMS sense embeddings (Loureiro and Jorge, 2019a) to disambiguate the target words. A simple approach of checking if the disambiguated senses are equal lead to competitive performance in the SemDeep-5 WiC challenge (Anke et al., 2019). SENSEMBERT (Scarlini et al., 2020a) and ARES (Scarlini et al., 2020b) embeddings, when used as features in a

BERT-based model, also achieve competitive results on the WiC task.

Our methods combine elements of both paradigms. We employ contextual embeddings in our proposed translation-based methods. However, we take the embeddings of the translations of the target words instead of the target words themselves. Similarly to WSD based approaches, our methods also analyze the common synsets of the focus tokens and their translations, with the goal of identifying a probable shared synset. The most similar prior work to our approach is that of Pesutto et al. (2020) at the graded word similarity task (Armendariz et al., 2020) of SemEval 2020, who propose a translation-based approach to evaluate the contextual similarity of a pair of words. They hypothesize that leveraging similarity information from more languages would allow greater accuracy. We follow a similar intuition in our work.

## 3 Theoretical Solution

We first present a theoretical solution, which provides the foundation for the development of our actual methods described in Section 4. We assume that the two source sentences  $S_1$  and  $S_2$  in each instance of the WiC task can be translated into any natural language as sentences  $T_1$  and  $T_2$ . Furthermore, we assume that the literal lexical translations  $t_1$  and  $t_2$  of the focus word  $s$  can be identified in  $T_1$  and  $T_2$ , respectively. For example, in Figure 1, the focus word  $s$  in the English sentences  $S_1$  and  $S_2$  is the noun *differential*, and word alignment identifies *écart* and *différentiel* as  $t_1$  and  $t_2$ . Note that the two translations may have the same POS and lemma, a scenario we denote as  $t_1 = t_2$ .

### 3.1 Substitution Test

Our theoretical solution is based on the notion of the linguistic *substitution test* for verifying the synonymy of senses (Hauer and Kondrak, 2020b), which takes as input two sentences *which differ only in a single word*, and returns TRUE if and



only if the two sentences have the same meaning. In other words, it decides whether the substitution of one word with another changes the meaning of the sentence. Note that this substitution test is not sufficient to decide the WiC task, as the input sentences for this task *share* a single word, rather than *differ* in a single word. The substitution test can be implemented by consulting a native speaker, or approximated by a computer program. In Section 4, we discuss an implementation based on contextual embeddings.

An example of a valid input to the substitution test would be the sentences *I work at the plant* and *I work at the factory*. For this input, the substitution test would return TRUE, since the word substitution does not change the meaning of the sentence. The sentences *I work at the plant* and *I work at the flower* would likewise constitute a valid input; however, given these sentences, the substitution test would return FALSE, since the sentences differ semantically.

### 3.2 Translation Criss-Cross

In order to apply the substitution test to an instance of the WiC task, we first translate the two source input sentences  $S_1$  and  $S_2$  into a target language, producing two target sentences  $T_1$  and  $T_2$ . We identify the two lexical translations  $t_1$  and  $t_2$  of the focus word  $s$  in  $T_1$  and  $T_2$ . Assuming that the translations are correct and literal, the senses of  $s$  in  $S_1$  and  $t_1$  in  $T_1$  will be synonymous, as well as the senses of  $s$  in  $S_2$  and  $t_2$  in  $T_2$ . If  $t_1$  and  $t_2$  have the same POS but different lemmas, we can replace  $t_1$  with  $t_2$  in  $T_1$  to produce a sentence  $T_1'$  which differs from  $T_1$  in a single word. The application of the substitution test to  $(T_1, T_1')$  returns TRUE if and only if the sense of  $t_2$  in  $T_1'$  is synonymous with the sense of  $s$  in  $S_1$ , which implies that, in addition to  $s$  and  $t_1$ , the multi-synset containing the sense of  $s$  in  $S_1$  must also include  $t_2$ .

Using our running example in Figure 1,  $T_1'$  would be created by replacing *écarts* with *différentiel* in  $T_1$ . This produces *les différentiel de taux d'intérêt croissant*, which, while not necessarily grammatical, can still be evaluated by the substitution test to decide whether the substitution alters the semantic content of the sentence. (Or, equivalently, whether *écart* and *différentiel* are synonymous in this particular context.)

We repeat the process with the roles of  $T_1$  and  $T_2$  reversed. That is, we construct  $T_2'$  by replacing

$t_2$  with  $t_1$  in  $T_2$  in order to verify whether the sense of  $t_1$  in  $T_2'$  is synonymous with the sense of  $s$  in  $S_2$ . If the substitution test returns FALSE for either of the two target sentence pairs, we can conclude that the two multi-synsets that correspond to the senses of  $s$  in  $S_1$  and  $S_2$  must be different. Therefore, this instance of the WiC task is resolved as FALSE. However, if the substitution test returns TRUE for both pairs of sentences, we cannot immediately resolve the instance of the WiC task, because there could exist two (or more) multi-synsets that all contain  $s$ ,  $t_1$ , and  $t_2$ . To complicate matters, this partial solution to the WiC task can only be applied if  $t_1$  and  $t_2$  have the same POS but different lemmas.

A complete theoretical solution can be obtained by considering translations in multiple languages. If the focus word  $s$  is not used in the same sense in  $S_1$  and  $S_2$ , we would expect that in *some* language, the translations  $t_1$  and  $t_2$  will be different *and* not mutually replaceable in both sentences. This expectation is consistent with the speculation of Palmer et al. (2007) that translation into a sufficiently large set of language will eventually lexicalize every sense distinction. It is also supported by the findings of Bao et al. (2021) who found no evidence for the existence of universal colexifications, that is, pairs of concepts that are expressed by the same word in every natural language.

### 3.3 Multi-Synset Intersection

For each language  $F_i$  in the set of all natural languages  $\mathcal{L}$ , let  $t_1^i$  and  $t_2^i$  be the lexical translations of the focus word  $s$  in the first and second input sentences, respectively. Let  $T$  be the set consisting of the focus word, and all its lexical translations; that is  $W = \{s\} \cup_{F_i} \{t_1^i, t_2^i\}$ . Assuming access to a perfect universal multi-wordnet, we define the set  $C$  to be the set of multi-synsets that contain all words in  $T$ .

The size of  $C$  provides clues to the resolution of the WiC task. We need to consider three cases:  $|C| = 0$ ,  $|C| = 1$ , and  $|C| \geq 2$ . With some caveats, these three cases roughly imply the following answers to the WiC task: FALSE, TRUE, and UNKNOWN, respectively. We discuss these three cases in turn.

If  $|C| = 0$ , then no single concept can be expressed by  $s$  and all its translations in  $T$ , according to the multi-wordnet. That is, there exist two translations of the focus word which cannot express the same concept, assuming the completeness of

the multi-wordnet. Therefore, the two focus tokens must correspond to distinct multi-synsets, implying FALSE.

If  $|C| = 1$ , there exists exactly one multi-synset that contains the focus word and all its translations. Therefore, it is possible, albeit not guaranteed, that the focus word in both source sentences is used in the sense that corresponds to that unique multi-synset. In order to be sure, we could apply the criss-cross method described in Section 3.2.

$|C| \geq 2$  would imply that there exist two concepts which are colexified (expressed by a single word) in all languages. Following Bao et al. (2021), we assume that universal collocations are at best extremely rare. Even if they exist at all, we could still apply the solution described in Section 3.2 to decide the WiC task. Of course, if we are considering translations into only a small number of languages, the possibility of  $|C| \geq 2$  is much more likely. In fact, we observe  $|C| = 3$  in our running example, because three different BabelNet multi-synsets contain the English focus word and its two French translations.

## 4 Methods

In this section we describe four methods based on the theoretical ideas in Section 3. All four methods rely on identifying lexical translations of the focus word in both source sentences. If the lexical translations cannot be recovered from the translated sentences for any of the target languages, all methods use the same backoff approach, which is to return FALSE for that test instance.

### 4.1 IDENT and CVAL

Our two simplest methods are IDENT and CVAL. IDENT is a baseline method which returns TRUE **iff** the lexical translations  $t_1$  and  $t_2$  have the same lemma and POS in all applicable target languages. CVAL is a method directly based on the cardinality of the set  $C$  as defined in Section 3.3. CVAL returns TRUE **iff** the translations of the focus word are identical in each language **and**  $|C| > 0$ .

### 4.2 Synonymy Check

We implement the substitution test as a heuristic *synonymy check* using dense contextualized embeddings. Such embeddings allow us to construct, for any word token in a given sentence, a vector in a continuous semantic space. The objective in designing such embeddings is that semantically

similar tokens should have similar vectors, commonly measured by cosine similarity. Additional technical details of the embeddings are provided in Section 5.

Given a pair of sentences which differ only in the substitution of single word, we obtain dense contextualized embeddings of the distinguishing word in each sentence. We then calculate the cosine similarity between the two embeddings. If the similarity is greater than a threshold tuned on a development set, this is taken as an indication that replacing one of the distinguishing words with the other does not alter the meaning of the sentence, as the replacement word has the same meaning as the original word. This implementation of the substitution test is used as a subroutine by our remaining two methods.

### 4.3 SUB and CSUB

The SUB method attempts to apply the synonymy check to each pair of translated sentences  $T_1$  and  $T_2$  in each target language, without referring to the  $|C|$  value. If the translations of the focus word in  $T_1$  and  $T_2$  differ, we create the sentences  $T'_1$  and  $T'_2$ , as described in Section 3.2, and apply the synonymy check to  $(T_1, T'_1)$  and  $(T_2, T'_2)$ . SUB returns TRUE if the synonymy check succeeds for all target languages for which the translations  $t_1$  and  $t_2$  can be identified. The synonymy check trivially succeeds if  $t_1$  and  $t_2$  have the same POS and lemma; intuitively, tokens which translate the same way are likely to have similar meanings. If either application of the synonymy check fails, SUB returns FALSE. In summary, this method is similar to the IDENT method, except that the synonymy check is applied if the translations differ.

CSUB combines CVAL with SUB. The only difference with the SUB method is that the synonymy check is not applied when  $|C| = 0$ . This is because the lack of any common multi-synset in a complete perfect multi-wordnet is theoretically sufficient to exclude the possibility of the two source focus tokens having the same sense.

## 5 Experiments

In this section, we describe the application of our methods to the English development and test sets. We begin by specifying various implementation details. Next, we describe our development experiments, including results and error analysis. Finally, we present our results on the test set. While our

method is, in theory, applicable to any language, and even to cross-lingual subtasks, we focus exclusively on the English monolingual subtask due to time and resource constraints.

### 5.1 Translation and Lemmatization

We use BabelNet (Navigli and Ponzetto, 2010, 2012) as our multi-wordnet; in particular, we make use of the BabelNet multi-synsets which are linked to Princeton Wordnet synsets. This allows us to exclude synsets that refer to named entities, rather than lexicalized concepts, to limit the impact of noise in BabelNet.

For translation, we use Google Translate, as it is fast and publicly available. In our analysis, we found the lexical translations obtained using Google Translate to be of generally high quality, which is important given our method’s dependence on machine translation. We use French, Italian, and Russian as our languages of translation. The choice of the translation languages is based on the languages selected for the shared task, and also on the BabelNet coverage. French and Russian are two of the languages covered by the shared task. On the other hand, Italian seems to have the best BabelNet coverage among the non-English languages.

For lemmatization, we use TreeTagger (Schmid, 1999, 2013), with pre-trained lemmatization models for the source and all target languages. We lemmatize the bitexts to improve the quality of the word alignment.

### 5.2 Word Alignment

Following lemmatization, we align each input sentence with its translation in each target language. To improve the quality of our unsupervised alignment, we obtain a large sentence-aligned parallel corpus (*bitext*) in the source and target languages. We then append to the bitext all of the lemmatized input sentences, and all of their lemmatized language translations. Finally, we apply an unsupervised knowledge-based alignment algorithm to the augmented bitext, and, for each sentence, identify the word or phrase in the translated sentence corresponding to the source focus word. Once each input sentence is aligned with its translation, we extract the lemmas aligned with each focus word token. These are the lexical translations of the focus word for this language.

To carry out the alignment, we use BabAlign (Luan et al., 2020), a state-of-the-art knowledge-based aligner. BabAlign leverages translation infor-

mation from BabelNet to create synthetic training data and post-process the alignment produced using a base unsupervised alignment method. Specifically, we use FastAlign (Dyer et al., 2013) as the base aligner. When aligning input sentences with translations, we concatenate the sentences and their translations with the OpenSubtitles bitext (Lison and Tiedemann, 2016) for the corresponding language pair. For each language pair, we use the first 1M sentences of the OpenSubtitles bitext.

### 5.3 Contextual Embeddings

To obtain contextual representations for the purposes of deciding the substitution check, we use BERT (Devlin et al., 2019), a deep neural architecture trained with the masked language model. We chose BERT because it has been proven to capture the semantics of a word in context (Coenen et al., 2019). The context is the sentence containing the focus word. Specifically, we use cased multilingual BERT to generate contextualized embedding of focus words by summing up the last four hidden layers of the BERT model. This choice was based on the results achieved by Devlin et al. (2019) in the named entity recognition task, and by Soler et al. (2019) in the SemDeep-5 WiC shared task.<sup>1</sup>

We use cased multilingual BERT embeddings with 768 dimensions, 12 layers, 12 attention heads, and 179M parameters. To implement the substitution check, we generate contextualized embeddings of the translations of the focus tokens, and their substitutes, by summing the last four hidden layers of the BERT model. Since BERT uses sub-tokens to generate embeddings, we analyzed the impact of two different sub-token selection techniques for predicting word similarity: using only the first sub-token, and using the mean over all the sub-tokens. In our development experiments, we found that the former yielded better results. Therefore, only the first sub-token is used to create contextualized embeddings for the substitution method.

### 5.4 Development Results

Table 1 shows the results of our development experiments. The baseline translation identity method IDENT does surprisingly well, outperforming both methods based on intersecting sets of multi-synsets, CVAL and CSUB. Indeed, these methods tend to suffer accuracy degradation as more languages of translation are added. We speculate that this is due

<sup>1</sup><https://www.dfki.de/declerck/semdeep-5/challenge.html>

Lang.	FR	IT	RU	ALL
IDENT	<b>59.6</b>	<b>58.1</b>	<b>57.1</b>	59.7
CVAL	58.9	57.6	54.3	55.5
SUB	59.3	58.0	55.6	<b>60.8</b>
CSUB	59.2	57.8	54.3	54.1

Table 1: MCL-WiC accuracy (%) on the En-En dev set with different methods and languages of translation.

to these methods being more vulnerable to noise (errors or omissions) in the multi-wordnet and in the extraction of lexical translations. However, the best performing method is SUB, which also shows improvement when combining all three languages of translation. Thus, it also shows the most promise for further improvement by adding additional languages.

Our error analysis suggests that there are three principal causes of errors. First, translation may be non-literal. For example, in one instance, the adverb “unevenly” is translated into French as the adjective “inégalé” (“unequal”), leading to a false negative. Second, distinct but synonymous translations may lead to false positives. In one instance, the focus word “stain” is translated as “souillé” in one sentence and “tachée” in the other. The focus tokens have distinct meanings, reflected in their distinct translations, “stain on a reputation” versus “stain on a surface”. However, the translations pass the BERT-based synonymy check, since they can be synonymous in some contexts. Finally, in some cases, distinct senses of a word may nevertheless translate the same way. For example, in one instance, the focus word “superior” was used in two distinct meanings. Both these meanings can be expressed by the French word “supérieur”, and indeed, “superior” was translated as “supérieur” in both sentences, resulting in a false positive.

### 5.5 Test Results and Discussion

Table 2 shows our results on the test data. Consistent with our development experiments, the SUB method achieves the best performance with the combination of all three languages. The IDENT method once again performs surprisingly well despite its simplicity, outperforming the more complex CVAL and CSUB methods. Different from the development experiments, when only one language of translation is used, Russian yields substantially better performance compared to French or Italian across all four methods, and Italian likewise yields

Lang.	FR	IT	RU	ALL
IDENT	55.8	<b>58.9</b>	<b>61.0</b>	61.1
CVAL	54.8	55.6	56.0	55.2
SUB	<b>56.1</b>	57.6	60.6	<b>63.2</b>
CSUB	55.2	55.2	55.8	55.7

Table 2: MCL-WiC accuracy (%) on the En-En test set with different methods and languages of translation.

better performance than French.

Table 3 gives additional details for the results of the SUB method. For each of the three languages, and the combination of all three, we provide the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), as well as the accuracy. We observe that using multiple languages of translation results in a substantial reduction in false positives, at the possible expense of an increase in false negatives, while maintaining an overall higher accuracy.

Lang.	TP	TN	FP	FN	Accuracy
FR	369	192	308	131	56.1
IT	376	200	300	124	57.6
RU	327	279	221	173	60.6
ALL	339	293	207	161	63.2

Table 3: Detailed breakdown of the results of our best performing method, SUB.

## 6 Conclusion

Overall, our results provide a solid proof-of-concept for the utility of multilingual translation for the WiC task. While not competitive with state-of-the-art supervised methods, our results empirically verify the hypothesis that translations convey semantic information, and that this phenomenon has applications in lexical semantics. The IDENT and SUB methods consistently benefit from translation into multiple languages, and this result generalizes to unseen test data.

## Acknowledgements

We thank the organizers of the shared task for their effort. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and the Alberta Machine Intelligence Institute (Amii).

## References

- Luis Espinosa Anke, Thierry Declerck, Dagmar Gromann, Jose Camacho-Collados, and Mohammad Taher Pilehvar. 2019. Proceedings of the 5th workshop on semantic deep learning (SemDeep-5). In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*.
- Carlos Santos Armendariz, Matthew Purver, Senja Polak, Nikola Ljubešić, Matej Uličar, Ivan Vulić, and Mohammad Taher Pilehvar. 2020. SemEval-2020 task 3: Graded word similarity in context. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 36–49.
- Hongchang Bao, Bradley Hauer, and Grzegorz Kondrak. 2021. On universal colexifications. In *Proceedings of the 11th Global Wordnet Conference (GWC2021)*, pages 1–7.
- Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda Vigas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of BERT. *arXiv preprint arXiv:1906.02715*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.
- Christiane Fellbaum. 1998. WordNet: An on-line lexical database and some of its applications. MIT Press.
- Goran Glavaš, Ivan Vulić, Anna Korhonen, and Simone Ponzetto. 2020. SemEval-2020 task 2: Predicting multilingual and cross-lingual (graded) lexical entailment. In *Proceedings of the 13th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.
- Bradley Hauer, Amir Ahmad Habibi, Yixing Luan, Arnob Mallik, and Grzegorz Kondrak. 2020. UALBERTA at SemEval-2020 task 2: Using translations to predict cross-lingual entailment. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 263–269, Barcelona (online). International Committee for Computational Linguistics.
- Bradley Hauer and Grzegorz Kondrak. 2020a. One homonym per translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7895–7902.
- Bradley Hauer and Grzegorz Kondrak. 2020b. Synonymy = translational equivalence. *arXiv preprint arXiv:2004.13886*.
- Bradley Hauer, Yixing Luan, and Grzegorz Kondrak. 2019. You shall know the most frequent sense by the company it keeps. In *2019 IEEE 13th International Conference on Semantic Computing (ICSC)*, pages 208–215. IEEE.
- Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 923–929. European Language Resources Association.
- Qianchu Liu, Diana McCarthy, and Anna Korhonen. 2020. Towards better context-aware lexical semantics: Adjusting contextualized representations through static anchors. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4066–4075.
- Daniel Loureiro and Alípio Jorge. 2019a. Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5682–5691, Florence, Italy. Association for Computational Linguistics.
- Daniel Loureiro and Alípio Jorge. 2019b. LIAAD at SemDeep-5 challenge: Word-in-Context (WiC). *arXiv preprint arXiv:1906.10002*.
- Yixing Luan, Bradley Hauer, Lili Mou, and Grzegorz Kondrak. 2020. Improving word sense disambiguation with translations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4055–4065, Online. Association for Computational Linguistics.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. SemEval-2021 task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 216–225, Uppsala, Sweden. Association for Computational Linguistics.
- Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.
- Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. 2007. Making fine-grained and coarse-grained sense distinctions, both manually and automatically. *Natural Language Engineering*, 13(2):137–163.

- Lucas RC Pessutto, Tiago de Melo, Viviane P Moreira, and Altigran da Silva. 2020. BabelEnconding at SemEval-2020 task 3: Contextual similarity as a combination of multilingualism and language models. *arXiv preprint arXiv:2008.08439*.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2018. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. *arXiv preprint arXiv:1808.09121*.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020a. SensEmbBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8758–8765.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020b. With more contexts comes better performance: Contextualized sense embeddings for all-round word sense disambiguation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3528–3539.
- Helmut Schmid. 1999. Improvements in part-of-speech tagging with an application to German. In *Natural language processing using very large corpora*, pages 13–25. Springer.
- Helmut Schmid. 2013. Probabilistic part-of-speech tagging using decision trees. In *New methods in language processing*, page 154.
- Aina Garí Soler, Marianna Apidianaki, and Alexandre Allauzen. 2019. LIMSI-MULTISEM at the IJCAI SemDeep-5 WiC challenge: Context representations for word usage similarity estimation. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 6–11.

# TransWiC at SemEval-2021 Task 2: Transformer-based Multilingual and Cross-lingual Word-in-Context Disambiguation

Hansi Hettiarachchi<sup>♡</sup>, Tharindu Ranasinghe<sup>§</sup>

<sup>♡</sup>School of Computing and Digital Technology, Birmingham City University, UK

<sup>§</sup>Research Group in Computational Linguistics, University of Wolverhampton, UK

`hansi.hettiarachchi@mail.bcu.ac.uk`

`tharindu.ranasinghe@wlv.ac.uk`

## Abstract

Identifying whether a word carries the same meaning or different meaning in two contexts is an important research area in natural language processing which plays a significant role in many applications such as question answering, document summarisation, information retrieval and information extraction. Most of the previous work in this area rely on language-specific resources making it difficult to generalise across languages. Considering this limitation, our approach to SemEval-2021 Task 2 is based only on pretrained transformer models and does not use any language-specific processing and resources. Despite that, our best model achieves 0.90 accuracy for English-English subtask which is very compatible compared to the best result of the subtask; 0.93 accuracy. Our approach also achieves satisfactory results in other monolingual and cross-lingual language pairs as well.

## 1 Introduction

Words' semantics have a dynamic nature which depends on the surrounding context (Pilehvar and Camacho-Collados, 2019). Therefore, the majority of words tends to be polysemous (i.e. have multiple senses). For few examples, words such as "cell", "bank" and "report" can be mentioned. Due to this nature in natural language, it is important to focus on word-in-context sense while extracting the meaning of a word which appeared in a text segment. Also, this is a critical requirement to many applications such as question answering, document summarisation, information retrieval and information extraction.

Word Sense Disambiguation (WSD)-based approaches were widely used by previous research to tackle this problem (Loureiro and Jorge, 2019; Scarlini et al., 2020). WSD associates the word in a text with its correct meaning from a predefined sense inventory (Navigli, 2009). As such

inventories, WordNet (Miller, 1995) and BabelNet (Navigli and Ponzetto, 2012) were commonly used. However, these approaches fail to generalise into different languages as these inventories are often limited to high resource languages. Targeting this gap, SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation is designed to capture the word sense without relying on fixed sense inventories in both monolingual and cross-lingual setting. In summary, this task is designed as a binary classification problem which predicts whether the target word has the same meaning or different meaning in different contexts of the same language (monolingual setting) or different languages (cross-lingual setting).

This paper describes our submission to SemEval-2021 Task 2 (Martelli et al., 2021). Our approach is mainly focused on transformer-based models with different text pair classification architectures. We remodel the default text pair classification architecture and introduce several strategies that outperform the default text pair classification architecture for this task. For effortless generalisation across the languages, we do not use any language-specific processing and resources. In the subtasks where only a few training instances were available, we use few-shot learning and in the subtasks where there were no training instances were available, we use zero-shot learning taking advantage of the cross-lingual nature of the multilingual transformer models.

The remainder of this paper is organised as follows. Section 2 describes the related work done in the field of word-in-context disambiguation. Details of the task data sets are provided in Section 3. Section 4 describes the proposed architecture and Section 5 provides the experimental setup details. Following them, Section 6 demonstrates the obtained results and Section 7 concludes the paper with final remarks and future research directions.

## 2 Related Work

**Unsupervised systems** Majority of the unsupervised WSD systems use external knowledge bases like WordNet (Miller, 1995) and BabelNet (Navigli and Ponzetto, 2012). For each input word, its correct meaning according to the context can be found using graph-based techniques from those external knowledge bases. However, these approaches are only limited to the languages supported by used knowledge bases. More recent works like Hettiarachchi and Ranasinghe (2020a); Ranasinghe et al. (2019a) propose to use stacked word embeddings (Akbik et al., 2018) obtained by general purpose pretrained contextualised word embedding models such as BERT (Devlin et al., 2019) and Flair (Akbik et al., 2019) for unsupervised WSD. Despite their ability to scale over different languages, unsupervised approaches fall behind supervised systems in terms of accuracy.

**Supervised systems** Supervised systems rely on semantically-annotated corpora for training (Raganato et al., 2017; Bevilacqua and Navigli, 2019). Early approaches were based on traditional machine learning algorithms like support vector machines (Iacobacci et al., 2016). With the word embedding-based approaches getting popular in natural language processing tasks, more recent approaches on WSD were based on neural network architectures (Melamud et al., 2016; Raganato et al., 2017). However, they rely on large manually-curated training data to train the machine learning models which in turn hinders the ability of these approaches to scale over unseen words and new languages. More recently, contextual representations of words have been used in WSD where the contextual representations have been employed for the creation of sense embeddings (Peters et al., 2018). However, they also rely on sense-annotated corpora to gather contextual information for each sense, and hence are limited to languages for which gold annotations are available. A very recent approach SensEmBERT (Scarlini et al., 2020) provide WSD by leveraging the mapping between senses and Wikipedia pages, the relations among BabelNet synsets and the expressiveness of contextualised embeddings, getting rid of manual annotations. However, SensEmBERT (Scarlini et al., 2020) only supports five languages making it difficult to use with other languages.

Considering the limitations of the above meth-

ods, in this paper we propose an approach which is based on general purpose transformer models and does not rely on external knowledge bases. Also, our approach shows strong few-shot/zero-shot learning performance removing the hurdle of having manually-curated training data for each language pair.

## 3 Data

The data set used for SemEval-2021 Task 2 is designed targeting a binary classification problem following Pilehvar and Camacho-Collados (2019). To preserve the multilinguality and cross-linguality of the task, five different languages: English, Arabic, French, Russian and Chinese have been considered for data set preparation. In the monolingual setting, per instance, a sentence pair written in the same language is provided with a targeted lemma to predict whether it has the same meaning (True) or different meanings (False) in both sentences. In the cross-lingual setting, each sentence pair is written in two different languages with the same prediction requirement. Few samples from the monolingual and cross-lingual data sets are shown in Table 1.

The monolingual data set covers the language pairs: en-en, ar-ar, fr-fr, ru-ru and zh-zh. For each language, 8-instance trial data sets with labels were provided to give an insight into the task. As training data, 8,000 labelled instances were provided only for the English language and as dev data, 1,000 labelled instances were provided per each language. To use with final evaluation, for each language, 1,000-instance test data sets were provided.

The cross-lingual data set covers the language pairs: en-ar, en-fr, en-ru and en-zh. Similar to the monolingual data set, 8-instance trial data sets with labels were provided for each language pair. However, no training or dev data sets were provided for the cross-lingual setting. To use with the final evaluation, 1,000-instance test data sets were provided per each language pair.

## 4 TransWiC Architecture

The main motivation behind the TransWiC architecture is the success transformer-based architectures had in various natural language processing tasks like offensive language identification (Ranasinghe and Hettiarachchi, 2020; Ranasinghe et al., 2019c; Pitenis et al., 2020), offensive spans identification (Ranasinghe and Zampieri, 2021a; Ranasinghe et al., 2021), language detection (Jauhiainen et al.,



	Lang.	Sentence 1	Sentence 2	Label
ML	fr-fr	la <b>souris</b> mange le fromage	le chat court après la <b>souris</b>	T
	en-en	In the private <b>sector</b> , activities are guided by the motive to earn money.	The volume V of the <b>sector</b> is related to the area A of the cap.	F
CL	en-fr	click the right <b>mouse</b> button	le chat court après la <b>souris</b>	F
	en-fr	Any alterations which it is proposed to make as a result of this review are to be <b>reported</b> to the Interdepartmental Committee on Charter Repertory for its approval.	Il a aussi été <b>indiqué</b> que, selon les dossiers médicaux, Justiniano Hurtado Torre était mort de maladie.	T

Table 1: Monolingual (ML) and cross-lingual (CL) sentence pair samples with targeted lemma (highlighted in red colour) and label (T:True, F:False). Lang. column represent the languages which are indicated using ISO 639-1 codes<sup>1</sup>

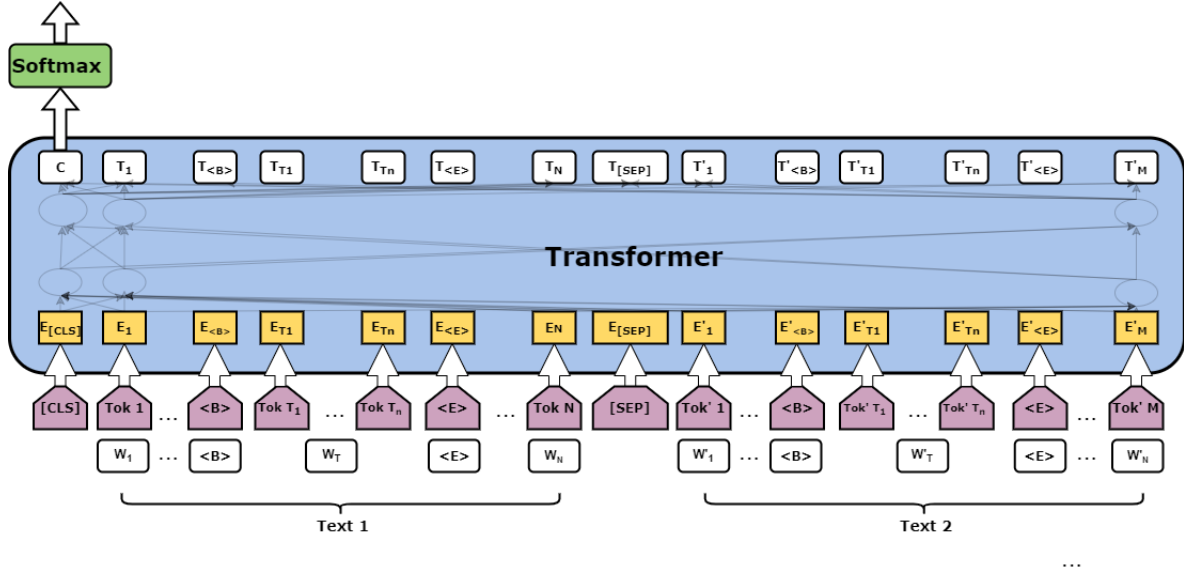


Figure 1: Default sentence pair classification architecture - ([CLS] Strategy).  $W_T$  is the target word.

2021) question answering (Yang et al., 2019) etc. Apart from providing strong results compared to RNN based architectures (Hettiarachchi and Ranasinghe, 2019; Ranasinghe et al., 2019c), transformer models like BERT (Devlin et al., 2019), XLM-R (Conneau et al., 2020) provide pretrained language models that support more than 100 languages. This is a huge benefit when compared to the models like SensEmBERT (Scarlini et al., 2020) which supports only five languages. Furthermore, multilingual and cross-lingual models like multilingual BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020) have shown strong transfer learning performance across scarce-resourced languages which can be useful in non-English monolingual subtasks where there are fewer training examples and cross-lingual subtasks where there are no training examples available (Ranasinghe and Zampieri, 2020, 2021b; Ranasinghe et al., 2020a). There-

fore we took the general purpose transformers like BERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020), reworked their sentence pair classification architecture with so called *strategies* described below to perform well in word-in-context disambiguation task.

**Preprocessing** As a preprocessing step we add two tokens to the transformer model’s vocabulary:  $\langle B \rangle$  and  $\langle E \rangle$ . We place them around the target word in both sentences. For example, the sentence "la souris mange le fromage" with the target word "souris" will be changed to "la  $\langle B \rangle$  souris  $\langle E \rangle$  mange le fromage".

- i [CLS] Strategy - This is the default sentence pair classification architecture with transformers (Devlin et al., 2019) where the two sentences are concatenated with a [SEP] token and passed through a transformer model. Then the

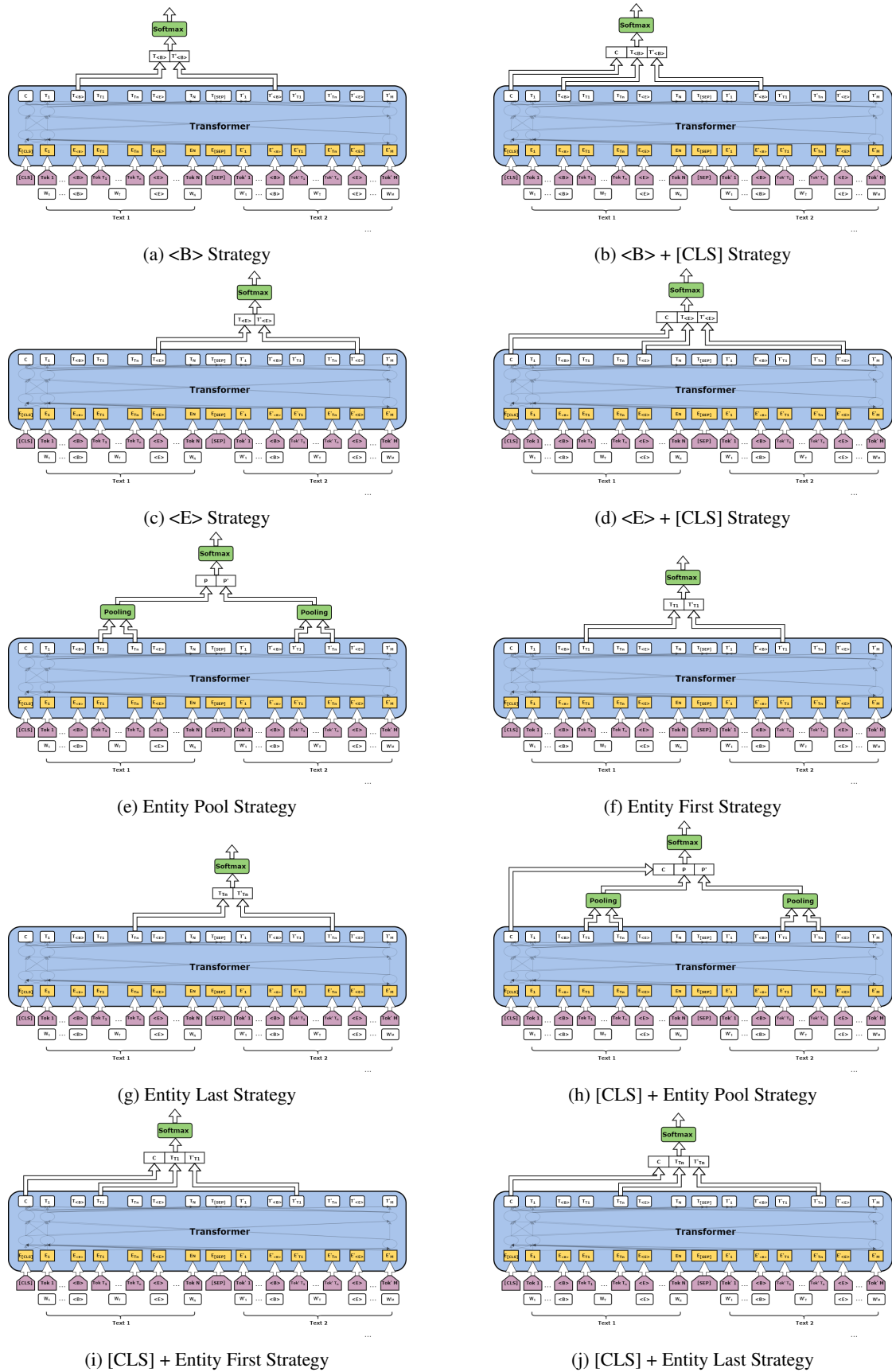


Figure 2: Strategies in the TransWiC Framework.  $W_T$  is the target word.

output of the [CLS] token is fed into a softmax layer to predict the labels (Figure 1).

- ii **<B> Strategy** - We concatenate the output of two <B> tokens of the two sentences and feed it into a softmax layer to predict the labels (Figure 2a).
- iii **<B> + [CLS] Strategy** - We concatenate the output of two <B> tokens of the two sentences with the [CLS] token and feed it into a softmax layer to predict the labels (Figure 2b).
- iv **<E> Strategy** - Output of the two <E> tokens of the two sentences are concatenated and feed into a softmax layer to predict the labels (Figure 2c).
- v **<E> + [CLS] Strategy** - We concatenate the output of two <E> tokens of the two sentences with the [CLS] token and feed it into a softmax layer to predict the labels (Figure 2d).
- vi **Entity Pool Strategy** - To effectively deal with rare words, transformer models use sub-word units or WordPiece tokens as the input to build the models (Devlin et al., 2019). Therefore, there is a possibility that one target word can be separated into several sub-words. In this strategy, we generate separate fixed-length embeddings for each target word by passing its sub-word outputs through a pooling layer. The pooled outputs are concatenated and fed into a softmax layer to predict the labels (Figure 2e).
- vii **Entity First Strategy** - Similar to the previous strategy, instead of using all the sub-words of the target word, we only use the output of the first sub-word in this strategy. We feed the concatenation of these outputs into a softmax layer to predict the labels (Figure 2f).
- viii **Entity Last Strategy** - Similar to the *Entity First Strategy* instead of the first sub-word, we use the last sub-word to represent the target word. We feed their concatenation into a softmax layer to predict the labels (Figure 2g).
- ix **[CLS] + Entity Pool Strategy** - We concatenate the pooled outputs generated by *Entity Pool Strategy* with the [CLS] token and feed it into a softmax layer to predict the labels (Figure 2h).

- x **[CLS] + Entity First Strategy** - Similar to the *[CLS] + Entity Pool Strategy*, instead of the pooled outputs, we concatenate the first sub-word output of the target words with [CLS] token and feed it into a softmax layer to predict the labels (Figure 2i).

- xi **[CLS] + Entity Last Strategy** - In this strategy, we concatenate the last sub-word output of the target words with [CLS] token and feed it into a softmax layer to predict the labels (Figure 2j).

## 5 Experimental Setup

This section describes the training data and hyperparameter configurations used during the experiments.

### 5.1 Training Configurations

**English-English** For the English-English sub-task, we performed training on the English-English training data for each strategy mentioned above. During the training process, the parameters of the transformer model, as well as the parameters of the subsequent layers, were updated. We used the saved model from a particular strategy to get predictions for the English-English test set for that particular strategy.

**Other Monolingual** Since there were less training data available for non-English monolingual datasets, we followed a *few-shot* learning approach mentioned in Ranasinghe et al. (2020c,b). When we are starting the training for non-English monolingual language pairs, rather than training a model from scratch, we initialised the weights saved from the English-English experiment. Then we performed training on the dev data for each language pair separately. Similar to English-English experiments, during the training process, the parameters of the transformer model, as well as the parameters of the subsequent layers, were updated.

**Crosslingual** Since there were no training data available for cross-lingual datasets, we followed a *zero-shot* approach for them. Multilingual and cross-lingual transformer models like multilingual BERT and XLM-R show strong cross-lingual transfer learning performance. They can be trained on one language; typically a resource-rich language and can be used to perform inference on another language. The cross-lingual nature of the transformer models has provided the ability to do this

(Ranasinghe et al., 2020c). Therefore, we used the models trained on the English-English dataset to get predictions for cross-lingual datasets.

## 5.2 Hyperparameter Configurations

We used a Nvidia Tesla K80 GPU to train the models. We divided the input dataset into a training set and a validation set using 0.8:0.2 split. We predominantly fine-tuned the learning rate and the number of epochs of the classification model manually to obtain the best results for the validation set. We obtained  $1e^{-5}$  as the best value for the learning rate and 3 as the best value for the number of epochs. We performed *early stopping* if the validation loss did not improve over 10 evaluation steps. The rest of the hyperparameters which we kept as constants are mentioned in the Appendix. When performing training, we trained five models with different random seeds and considered the majority-class self ensemble mentioned in Hettiarachchi and Ranasinghe (2020b) to get the final predictions.

## 6 Results and Evaluation

Organisers used the accuracy as the evaluation metric as shown in Equation 1 where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Since there were less or no training data available for other monolingual and cross-lingual settings, we trained and evaluated models for each of our strategies using English-English training and dev sets. Then the best models are picked to use with few-shot and zero-shot learning approaches. We report the results obtained by English-English evaluation in Table 2. In the BERT column, we report the results of the bert-large-cased model while in the XLM-R column, we report the results of the xlm-r-large model.

As shown in Table 2, some strategies outperformed the default sentence pair classification architecture. Among all experimented strategies <B> + [CLS] strategy performed best. Usually, multilingual transformer models like XLM-R do not outperform the language-specific transformer models. Surprisingly, in this task XLM-R models outperform bert-large models. We selected three best performing models for the submission; XLM-R

Strategy	BERT	XLM-R
CLS	0.8350	0.7860
<B>	0.8450	0.8750 <sup>‡</sup>
<B> + CLS	0.8590	<b>0.8810<sup>‡</sup></b>
<E>	0.6672	0.5590
<E> + CLS	0.6982	0.5630
Entity Pool	0.8420	0.8521
Entity First	0.8390	0.8462
Entity Last	0.8550	0.8660
CLS + Entity Pool	0.8570	0.8700 <sup>‡</sup>
CLS + Entity First	0.8540	0.8580
CLS + Entity Last	0.8568	0.8610

Table 2: TransWiC accuracy in English-English dev set for each strategy. Best is in Bold. Submitted systems are marked with ‡

<B> + [CLS], XLM-R <B> and XLM-R [CLS] + Entity Pool.

Since multilingual models provided the best results for the English-English dataset, it provided an additional advantage as they can be used directly in other language pairs too as mentioned in Section 5. For other language pairs, we did not perform any evaluation due to the lack of data availability. We trusted the cross-lingual performance of XLM-R and used the best three models of the English-English experiment. For the rest of the monolingual pairs, we used the few-shot learning approach using the given dev sets and for the cross-lingual pairs, we used the zero-shot learning approach mentioned in Section 5.

We report the results we got for the test set in Table 3. According to the results, <B> + [CLS] strategy performs best in all the language pairs except Ar-Ar, where <B> strategy outperforms <B> + [CLS] strategy. When compared to the best models submitted to each language pair, our approach shows very competitive results in the majority of the monolingual language pairs. However, we believe that the cross-lingual performance of our methodology should be improved. Nonetheless, we believe that as a methodology that did not use any language-specific resources and did not see any language-specific data, the results are at a satisfactory level.

## 7 Conclusions

In this paper, we presented our approach for tackling the SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation. We use the pretrained transformer models and re-

Strategy	Monolingual					Crosslingual				
	en-en	ar-ar	fr-fr	ru-ru	zh-zh	en-ar	en-fr	en-ru	en-zh	
I	<B> + [CLS]	<b>0.9040</b>	0.7800	<b>0.7970</b>	<b>0.7610</b>	<b>0.6210</b>	<b>0.6690</b>	<b>0.5860</b>	<b>0.6900</b>	<b>0.7640</b>
	<B>	0.8980	<b>0.7980</b>	0.7760	0.7160	0.6090	0.6260	0.5850	0.6770	0.7280
	[CLS] + Entity Pool	0.8400	0.7621	0.7321	0.6954	0.5880	0.5921	0.5572	0.6561	0.7002
II	Best System	0.9330	0.8480	0.8750	0.8740	0.9100	0.8910	0.8910	0.8940	0.9120

Table 3: Row I shows the accuracy scores for the test set with strategies submitted. Best results for each language pair with our strategies are in bold. Row II shows the accuracy scores for the test set with the best system submitted for each language pair.

model the sentence pair classification architecture for this task with several strategies. Our best strategies outperform the default sentence pair classification setting for English-English. For other monolingual language pairs, we use the few-shot learning approach while for cross-lingual language pairs we use the zero-shot approach. Our results are compatible with the best systems submitted for each language pair and are at a satisfactory level given the fact that we did not use any language-specific processing nor resources.

As future work, we would be looking to improve our results more with new strategies. We would like to experiment with whether adding language-specific processing and resources would improve the results. We are keen to add different neural network architectures like Siamese transformer networks (Reimers and Gurevych, 2019) that perform well in sentence pair classification tasks (Ranasinghe et al., 2019b; Mueller and Thyagarajan, 2016) to the TransWiC framework. Furthermore, we are hoping to work in a multi-task environment and experiment whether transfer learning from a similar task like semantic textual similarity (Cer et al., 2017) would improve the results for this task.

## References

- Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. [Pooled contextualized embeddings for named entity recognition](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. [Contextual string embeddings for sequence labeling](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Michele Bevilacqua and Roberto Navigli. 2019. [Quasi bidirectional encoder representations from transformers for word sense disambiguation](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 122–131, Varna, Bulgaria. INCOMA Ltd.
- Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. [SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2019. [Emoji powered capsule network to detect type and target of offensive posts in social media](#). In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 474–480, Varna, Bulgaria. INCOMA Ltd.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2020a. [BRUMS at SemEval-2020 task 3: Contextualised embeddings for predicting the \(graded\) effect of](#)

- context in word similarity. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 142–149, Barcelona (online). International Committee for Computational Linguistics.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2020b. **InfoMiner at WNUT-2020 task 2: Transformer-based covid-19 informative tweet extraction**. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 359–365, Online. Association for Computational Linguistics.
- Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. **Embeddings for word sense disambiguation: An evaluation study**. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany. Association for Computational Linguistics.
- Tommi Jauhiainen, Tharindu Ranasinghe, and Marcos Zampieri. 2021. Comparing approaches to dravidian language identification. In *Proceedings of the 7th Workshop on NLP for Similar Languages, Varieties and Dialects*.
- Daniel Loureiro and Alipio Jorge. 2019. **Liaad at semdeep-5 challenge: Word-in-context (wic)**. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 1–5.
- Federico Martelli, Najla Kalach, Gabriele Tola, and Roberto Navigli. 2021. **SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC)**. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*.
- Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. **context2vec: Learning generic context embedding with bidirectional LSTM**. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.
- George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.
- Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. *AAAI’16*, page 2786–2792. AAAI Press.
- Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.
- Roberto Navigli and Simone Paolo Ponzetto. 2012. **Babelnet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network**. *Artificial Intelligence*, 193:217–250.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. **Deep contextualized word representations**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. **Wic: the word-in-context dataset for evaluating context-sensitive meaning representations**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273.
- Zesis Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. **Offensive language identification in Greek**. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5113–5119, Marseille, France. European Language Resources Association.
- Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. 2017. **Neural sequence learning models for word sense disambiguation**. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, Copenhagen, Denmark. Association for Computational Linguistics.
- Tharindu Ranasinghe, Sarthak Gupte, Marcos Zampieri, and Ifeoma Nwogu. 2020a. **Wlv-rit at hasoc-dravidian-codemix-fire2020: Offensive language identification in code-switched youtube comments**. In *In Proceedings of the 12th annual meeting of the Forum for Information Retrieval Evaluation*.
- Tharindu Ranasinghe and Hansi Hettiarachchi. 2020. **BRUMS at SemEval-2020 task 12: Transformer based multilingual offensive language identification in social media**. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1906–1915, Barcelona (online). International Committee for Computational Linguistics.
- Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2019a. **Enhancing unsupervised sentence similarity methods with deep contextualised word representations**. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 994–1003, Varna, Bulgaria. INCOMA Ltd.
- Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2019b. **Semantic textual similarity with Siamese neural networks**. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 1004–1011, Varna, Bulgaria. INCOMA Ltd.
- Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2020b. **TransQuest at WMT2020:**

**Sentence-level direct assessment.** In *Proceedings of the Fifth Conference on Machine Translation*, pages 1049–1055, Online. Association for Computational Linguistics.

Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. 2020c. **TransQuest: Translation quality estimation with cross-lingual transformers.** In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5070–5081, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Tharindu Ranasinghe, Diptanu Sarkar, Marcos Zampieri, and Alex Ororbia. 2021. **WLV-RIT at SemEval-2021 Task 5: A Neural Transformer Framework for Detecting Toxic Spans.** In *Proceedings of SemEval*.

Tharindu Ranasinghe and Marcos Zampieri. 2020. **Multilingual offensive language identification with cross-lingual embeddings.** In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5838–5844, Online. Association for Computational Linguistics.

Tharindu Ranasinghe and Marcos Zampieri. 2021a. **MUDES: Multilingual Detection of Offensive Spans.** In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*.

Tharindu Ranasinghe and Marcos Zampieri. 2021b. **Multilingual Offensive Language Identification for Low-resource Languages.** *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*.

Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. 2019c. **BRUMS at HASOC 2019: Deep learning models for multilingual hate speech and offensive language identification.** In *Proceedings of the 11th annual meeting of the Forum for Information Retrieval Evaluation*.

Nils Reimers and Iryna Gurevych. 2019. **Sentence-BERT: Sentence embeddings using Siamese BERT-networks.** In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020. **SensEmBERT: Context-Enhanced Sense Embeddings for Multilingual Word Sense Disambiguation.** *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8758–8765.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. **End-to-end open-domain question answering with**

**BERTserini.** In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

## A Appendix

A summary of hyperparameters and their values used to obtain the reported results are mentioned in Table 4. The optimised hyperparameters are marked with ‡ and their optimal values are reported. The rest of the hyperparameter values are kept as constants.

Parameter	Value
learning rate‡	$1e^{-5}$
number of epochs‡	3
adam epsilon	$1e^{-8}$
warmup ration	0.1
warmup steps	0
max grad norm	1.0
max seq. length	120
gradient accumulation steps	1

Table 4: Hyperparameter specifications

# LIORI at SemEval-2021 Task 2: Span Prediction and Binary Classification approaches to Word-in-Context Disambiguation

**Adis Davletov**

RANEPA, Moscow, Russia  
Lomonosov Moscow State University, Moscow, Russia  
davletov-aa@ranepa.ru

**Nikolay Arefyev**

Lomonosov Moscow State University, Moscow, Russia  
Samsung Research Center Russia, Moscow, Russia  
HSE University, Moscow, Russia  
nick.arefyev@gmail.com

**Denis Gordeev**

RANEPA, Moscow, Russia  
gordeev-di@ranepa.ru

**Alexey Rey**

RANEPA, Moscow, Russia  
rey-ai@ranepa.ru

## Abstract

This paper presents our approaches to SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation task. The first approach attempted to reformulate the task as a question answering problem, while the second one framed it as a binary classification problem. Our best system, which is an ensemble of XLM-R based binary classifiers trained with data augmentation, is among the 3 best-performing systems for Russian, French and Arabic in the multilingual subtask. In the post-evaluation period, we experimented with batch normalization, subword pooling and target word occurrence aggregation methods, resulting in further performance improvements.

## 1 Introduction

In the Semeval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation task, the participants were asked to classify whether the target word, occurring in two sentences (sentence1 and sentence2), is used in the same or in a different meaning. The two sentences could be in the same language or different languages. There were two subtasks:

- **Multilingual Word-in-Context Disambiguation**, where both sentences were in the same language, either Arabic, Chinese, English, French or Russian
- **Cross-lingual Word-in-Context Disambiguation**, where the first sentence was in

English and the second one was either in Arabic, Chinese, French, or Russian.

More detailed information regarding the task is provided by [Martelli et al. \(2021\)](#).

We participated in both tracks and experimented with two approaches<sup>1</sup>. The first approach fine-tunes XLM-R ([Conneau et al., 2020a](#)) as a question answering system searching in the second sentence for a word with the same meaning as the target word in the first sentence. The second approach fine-tunes XLM-R as a binary classifier, and ensembles several such classifiers. Also, we used data augmentation to double the number of training examples. This second approach took the 2nd place for the monolingual subtask in Arabic and the 3rd place for the monolingual subtask in French and Russian. In the cross-lingual subtask, the system ranked 6th for French and Arabic. The same system was applied to all subtasks and languages.

During the post-evaluation period, we performed thorough experiments with our system. We compared different subword pooling methods, including mean, max, first pooling and their combinations, and found that combinations do not help and mean pooling is overall the best choice. Unlike pooling, instead of a simple concatenation of contextualized embeddings for the target word occurrences, it is helpful to combine their difference and normalized component-wise product. Finally, we found it beneficial to add a batch normalization

<sup>1</sup><https://github.com/davletov-aa/mcl-wic>



layer before feeding those vectors into the classification head.

## 2 Related Work

Word Sense Disambiguation (WSD) is the task of associating the occurrence of a word in a text with its correct meaning from a predefined inventory of senses (Navigli, 2009; Scarlini et al., 2020a). Word-in-Context Disambiguation is a new declination of WSD aiming to evaluate the ability of modern language models to accurately represent context-sensitive words (Pilehvar and Camacho-Collados, 2019; Scarlini et al., 2020b). Its advantage is that it does not rely on pre-defined sense inventories. Because Word Sense Disambiguation relies on world knowledge for successful solving (Navigli, 2009), modern large pre-trained models show promising results in solving this task.

Among such works, we can mention ARES (Scarlini et al., 2020b). ARES is a semi-supervised approach for creating sense embeddings. The authors use BERT and UKB (Agirre et al., 2014) to find contexts that are similar to each other and link them to meanings in WordNet (Miller et al., 1990). Then, they enrich synset contexts with collocational information from SyntagNet (Maru et al., 2019). Finally, they enrich SemCor (Miller et al., 1993) contexts and WordNet glosses to create sense-level representations. ARES performs better than models with a comparable number of parameters such as BERT or RoBERTa (Liu et al., 2019).

However, there has been substantial progress in the field of language modelling since BERT first appeared. Many researchers have noticed that BERT is undertrained and that training it longer and on more data, increases the model performance. Among such new models, we may name XLM-RoBERTa (XLM-R) (Conneau et al., 2020a). XLM-R, as well as BERT, is based on a Transformer model (Vaswani et al., 2017). XLM-R uses masked language modelling objective (Devlin et al., 2018; Lample and Conneau, 2019) for model training, where some tokens are replaced with a special "[MASK]" token and the model is to restore the masked tokens. XLM-R was trained on a cleaned two-terabyte CommonCrawl Corpus in 100 languages.

A new promising approach to language task modelling is treating any natural language task as a question answering problem. Among such works, we can mention (Cohen et al., 2020) where the au-

Set	Pos	Neg
train-en-en	4000	4000
dev-en-en	500 (0)	500 (0)
dev-ar-ar	500 (349)	500 (351)
dev-ru-ru	500 (337)	500 (363)
dev-fr-fr	500 (366)	500 (334)
dev-zh-zh	500 (323)	500 (377)
trial-xx-xx	x (x)	y (y)

Table 1: Statistics of the data provided by organizers. The numbers in brackets show the portion used as training examples. In trial set there were up to 8 examples for each of 9 multilingual and cross-lingual sets.

thors restructured relation classification as a Question Answering (QA) like span prediction problem. It allowed them to get state-of-the-art results for TACRED and SemEval 2010 task 8 datasets. We decided to adopt a similar approach to the task of word sense disambiguation.

## 3 Submitted Systems Description

Our systems are based on XLM-RoBERTa (XLM-R) model Conneau et al. (2020b). We used XLM-R large model as a backbone in all our submissions but switched to XLM-R base for some of the post-evaluation experiments. Two model training scenarios have been tested. In the first case (AG), due to the symmetric nature of the dataset, we decided to augment the dataset and flip the first and the second sentences. In the second case (MTL), multi-task learning was applied. More detailed descriptions are provided in the following sections. We used transformers library (Wolf et al., 2019).

### 3.1 Data

In all our experiments we used only the datasets provided in the shared task. For training, we employed the whole English train set, 70% of the development sets for other languages and all the trial data. The remaining data were used to select hyperparameters and do early stopping. We employed a lexical split resulting in different target words for training and validation. Table 1 presents detailed statistics. Optionally, in systems with **AG** suffix, train and test time data augmentation was performed by swapping sentences in each example to double the amount of data. If the predictions for symmetric examples were conflicting with each other we assumed the prediction is negative.

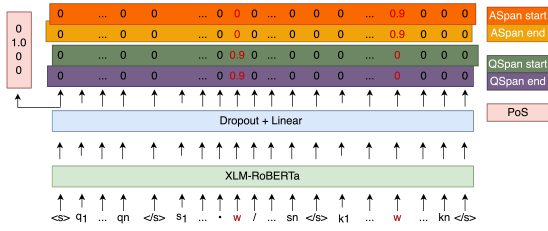


Figure 1: QA-based model architecture

### 3.2 QA Systems

Inspired by the work of Cohen et al. (2020), in our preliminary experiments we tried to solve MCLWIC in Question Answering (QA) task manner, where we predict the start and end positions (the span) of the answer in a given text.

Given the target word  $w$  and a pair of tokenized sentences  $[s_1 \dots w \dots s_n]$  and  $[k_1 \dots w \dots k_n]$ , we form the following input to XLM-R model with marked by  $\bullet$  and  $/$  symbols target word in the first sentence:

$[CLS]$  Find the same sense of the marked word  $[EOS]$   $s_1 \dots \bullet w \dots s_n [EOS]$   $k_1 \dots w \dots k_n [EOS]$ . We tokenize target word in context and its left and right contexts for both sentences separately.

The architecture of our QA system could be seen in Figure 1. We predict the span  $A$  (answer) of the target word in the second sentence if it is used in the same meaning as in the first sentence and the span of the  $[CLS]$  token otherwise. Also, we additionally predict the span of  $Q$  (question) of the target word in the first sentence. We use a dropout layer followed by a linear layer over XLM-R output  $o_i$  from the last layer at timesteps  $i$  to predict the probability that  $o_i$  is the start or the end of the spans  $Q$  and  $A$ .

As for each target word we had its part of speech label (PoS), we decided to predict it using a linear layer over the output corresponding to  $[CLS]$  token from the last layer of XLM-R.

During the training process, we optimize the weighted sum of cross-entropy losses of  $A$  span,  $Q$  span, and PoS predictions. And as the corresponding weights, we take the softmax over the learnable weights' vector  $V \in R^3$ .

We fine-tuned the models in the settings from Table 2. Four times per training epoch we were validating our models and saving the best one. During the inference we assumed the positive answer if the model predicted possible span  $A$  that satisfied conditions  $A_{start} < A_{end}$  and  $A_{start} > PrefixQuestion$ . We did not try to train QA sys-

Hyperparameter	Value
weight decay	0.1
warmup_proportion	0.1
dropout	0.1
learning rate	1e-4
learning_rate_scheduler	linear_warmup
optimizer	Adam
epochs	50
batch samples	64
max_sequence_length	256
max_gradient_norm	1.0

Table 2: Training hyperparameters of MTL-EN and MTL-XX systems, submitted to the competition

tems with symmetric data augmentation.

Further, we will be referring to the model validated on the English development set as the MTL-EN model. And as MTL-XX we will be referring to the models validated on one of the remaining development sets for the Russian, Arabic, French and Chinese languages.

### 3.3 BC Systems

Along with QA models, we tried a more traditional and straightforward approach of fine-tuning XLM-R as a binary classifier (BC).

So, given the target word  $w$  and a pair of tokenized sentences  $[s_1 \dots w \dots s_n]$  and  $[k_1 \dots w \dots k_n]$  we formed the following input example to XLM-R model:

$[CLS]$   $s_1 \dots w \dots s_n [EOS]$   $k_1 \dots w \dots k_n [EOS]$ . The sentences were tokenized the same way as in QA models.

We feed it to XLM-R and pool outputs  $o_s$  and  $o_k$  from the last layer from the subwords corresponding to the target word in two sentences. In our submissions we either took the output from the **first** subword, or used **max** pooling. In the post-evaluation, we also tried **mean** pooling and found, that it consistently provides the best results. Then we tried concatenating it with first (**mf**) or max (**mm**) pooled vectors, as well as both of them (**mmf**). Finally, we tried concatenating min, max and mean pooled outputs (**mmm**).

After obtaining fixed-sized representations of the first and the second target word occurrence, we concatenate them and feed them to the binary classifier, which is the sequence of dropout, linear, tanh, dropout and linear layers. The architecture of the model is depicted in Figure 2.

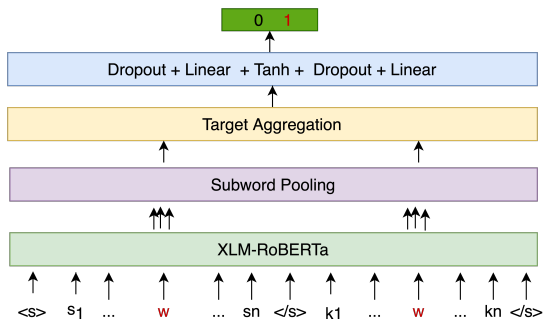


Figure 2: BC-based model architecture

In the post-evaluation period we tried replacing concatenation (**concat**) with alternative aggregation techniques. First, we tried using component-wise difference (**diff**) or multiplication (**mul**) with an optional normalization of word occurrence vectors (**mulnorm**). Then we tried combining representations obtained with different aggregation techniques by concatenating them. We denote those combinations by letter sequences, where **c** stands for the concatenation of the first and the second vector, **d** for their difference and **m** for their component-wise product. The inputs to each of those operations can be optionally normalized, which is denoted by **n** after the corresponding operation. For instance, **dmn** means that we concatenate the difference of non-normalized and the product of normalized vectors. Also in the post-evaluation, we found it beneficial to apply batch normalization before feeding aggregated representations into the classification head.

During training, we applied 2-class softmax and optimized the cross-entropy loss. We fine-tuned BC models using almost the same settings as for QA models. Here, we used the constant learning rate of  $1e-5$  with linear warmup during the first 10% of training steps. During post-evaluation, we added linear learning rate decay.

Our submitted BC systems are ensembles of these three models: first, first-AG and max-AG differing by subwords pooling strategy and by use of data augmentation. We would be referring to the ensemble of these models validated on the English dev set as ENS-EN and as ENS-XX for an ensemble of models validated on corresponding dev sets.

## 4 Experiments and Results

As our submissions showed us that BC models perform much better than QA models, in our post-

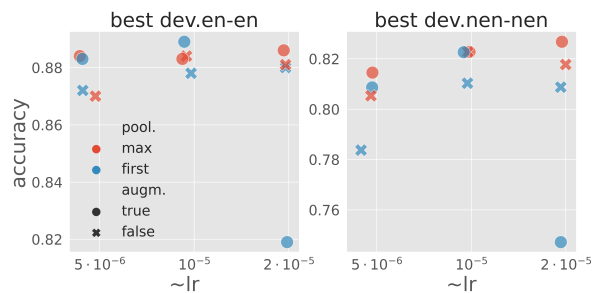


Figure 3: Data augmentation effect on xlmr.large performance



Figure 4: Comparison of target aggregation methods for xlmr.base (mean pooling, no batchnorm)

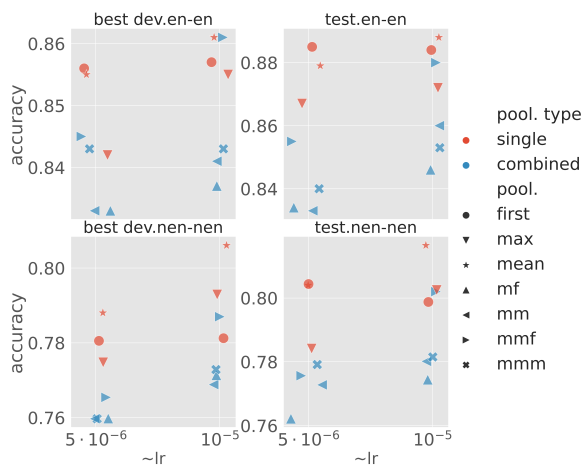


Figure 5: Comparison of subword pooling methods for xlmr.base (dmn agg. with batchnorm)

evaluation experiments we focused on them. In the following experiments, we report the best accuracy obtained during training on the English development set (*best dev.en-en*) and the best average accuracy on all non-English development sets (*best dev.nen-nen*) from our own split. For those epochs where the best dev set accuracy is achieved,

we report accuracy on the official English test set (*test.en-en*) and averaged over official non-English test sets (*test.nen-nen*). Due to space limitations, we report the results on all test sets only for our best models in Table 3.

In the first experiment, we trained the submitted models with and without data augmentation. Figure 3 shows that the augmentation never decreases performance, at least if the learning rate is selected properly. Thus, we always performed data augmentation in the following experiments.

Figure 4 compares target aggregation techniques. The concatenation of the difference of non-normalized vectors and component-wise product of normalized vectors (**dmn**) proved to outperform all other methods by a large margin, especially when the learning rate is properly selected. Thus, we used this technique for the following experiments. A simple concatenation of target embeddings, which was used in our submissions, is more than 3% worse and is often outperformed by the difference or component-wise product.

Since concatenating normalized and non-normalized vectors can make training difficult, we decided to apply batch normalization before feeding those vectors into the classification head. Figure 6 shows that batch normalization does not improve results for English much, but considerably improves the average performance on other languages. This is probably due to the fact, that the overwhelming majority of training examples are in English.

Also, from figure 6 we notice that for English different poolings give similar performance. For other languages, first pooling is a bad option. We hypothesise that this results from non-English words being split into sub-word tokens more frequently. Mean pooling consistently outperforms other poolings. Figure 5 additionally compares combinations of different subword poolings. However, those combinations did not improve results compared to single mean pooling.

Finally, we estimated how much the additional multilingual training data help compared to using only English training examples and counting on cross-lingual zero-shot transfer. In table 3 we denote xlmr.base models fine-tuned only on English train and trial data as **enonly**. We see that including non-English examples into the training set improves the results by 1.5-3% for multilingual and even more for cross-lingual scenarios. Surprisingly,

it also gives some improvement for English.

Table 3 summarizes the results of our submitted systems and the following post-evaluation experiments. During the evaluation period, we made a total of 4 submission attempts, two for the Question Answering based approach and two for the binary classifier approach. During training the best checkpoint was selected either individually for each language using corresponding dev set accuracy (XX), or by English dev set accuracy (EN). We see that the first approach to the task (MTL-EN, MTL-XX) shows much worse results compared to the second one (ENS-EN, ENS-XX). For the second approach, we submitted two ensembles consisting of three models shown in the same Table 3. As expected, ensembling the models helped to improve the results greatly.

As we figured out that **dmn** target aggregation and **mean** subword pooling performs significantly better compared to other variants for XLMR.base model, we trained XLMR.large version with hyperparameters from the best XLMR.base model. The results of the models validated either by score on English dev set (EN), or by the average score for non-English dev sets (nEN), or by scores on each dev set individually (XX), are shown in the third group of results in the Table3. We see that these models outperform any single model from the evaluation phase for all multilingual subtask’s test sets and test.en-ar set from cross-lingual subtask.

And lastly, we report the results for an ensemble of three XLMR.large models: two mean-dmn models trained with learning rates  $1e-5$  and  $2e-5$  and one mean-cnmm model trained with learning rate equal to  $1e-5$ . We see that using ensemble of models with new subword pooling and target aggregation techniques helps us to improve our official results from competition. We improved our results for test.ru-ru (3 → 2), test.fr-fr (2 → 1), test.en-en (15 → 12), test.zh-zh (21 → 17), test.en-ar (6 → 4) and test.en-zh (17 → 12) sets.

## 5 Conclusion

In this paper, we have described our approach to SemEval-2021 Task 2. We tried treating Word-in-Context Disambiguation as question answering and binary classification problems. In our case, binary classification turned out to be a more promising approach. Also, we found that mean pooling over subwords is the best option, batch normalization helps when added before classification head, and concate-

Model	ar-ar	ru-ru	fr-fr	en-en	zh-zh	en-ar	en-ru	en-fr	en-zh
our submissions: question answering based									
MTL-EN	73.9	75.0	77.9	84.7	77.7	63.1	66.8	69.8	69.3
MTL-XX	77.0	76.2	73.9	84.7	76.5	–	–	–	–
our submissions: binary classifier based									
ENS-EN	84.6(2)	85.3(10)	86.4(3)	91.1(15)	83.9(21)	86.5(6)	87.0(8)	87.2(6)	86.0(17)
ENS-XX	83.8(8)	86.6(3)	86.3(4)	91.1(15)	83.5(22)	–	–	–	–
first-concat-EN	83.5	84.2	84.8	90.0	82.2	85.4	86.4	86.4	84.4
first-concat-XX	82.0	84.6	84.8	90.0	82.1	–	–	–	–
first-concat-noAG-EN	82.3	82.5	85.4	90.8	81.4	84.9	85.7	86.1	85.8
first-concat-noAG-XX	82.9	84.0	84.9	90.8	80.9	–	–	–	–
max-concat-EN	83.2	85.8	84.1	89.8	84.5	85.7	82.6	84.2	81.9
max-concat-XX	83.6	83.3	84.7	89.8	84.9	–	–	–	–
post-evaluation results: xlmr.large, mean-dmn,lr=1e-5									
XX	84.0	86.1	84.5	90.7	83.5	–	–	–	–
EN	83.6	86.4	85.8	90.7	84.7	86.3	85.4	85.5	85.5
nEN	84.2	84.7	85.2	89.9	84.3	84.7	84.2	83.9	83.7
post-evaluation results: xlmr.large, mean-dmn,lr=2e-5 + mean-cnmm,lr=2e-5 + mean-dmn,lr=1e-5									
ENS-EN	84.6(2)	87.0(2)	87.5(1)	91.4(12)	84.8(17)	87.6(4)	86.2(12)	86.2(7)	87.1(12)
post-evaluation results: xlmr.base, mean-dmn									
XX	83.3	80.7	82.8	88.8	81.3	–	–	–	–
enonly-XX	80.5	79.4	80.7	87.1	80.1	–	–	–	–
EN	78.1	80.9	82.8	88.8	81.9	78.8	82.2	82.1	83.6
enonly-EN	81.9	78.6	80.7	87.1	79.6	75.5	79.5	76.7	73.4
nEN	82.1	80.7	83.4	89.1	81.3	80.7	82.4	82.7	79.9
enonly-nEN	81.3	79.4	81.4	87.8	81.0	74.3	77.9	75.9	72.5

Table 3: Results on all cross-lingual and monolingual test sets. XX denotes models validated on corresponding dev sets. For instance, XX model’s result for ru-ru set was obtained by model validated on dev.ru-ru set. nEN denotes models validated by averaged scores for non-English dev sets and EN denotes the ones validated on the English dev set. During the evaluation period we submitted MTL-EN, MTL-XX, ENS-EN and ENS-XX models’ predictions.

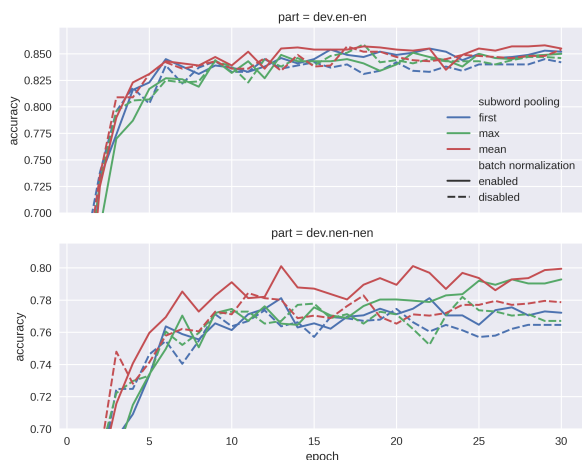


Figure 6: Effect of batch normalization on xlmr.base

nation of target embeddings is outperformed by the combination of the difference and the product of normalized embeddings.

## References

Eneko Agirre, Oier Lopez de Lacalle, and Aitor Soroa. 2014. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84.

Amir DN Cohen, Shachar Rosenman, and Yoav Goldberg. 2020. Relation Extraction as Two-way Span-Prediction. *arXiv preprint arXiv:2010.04829*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020a. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Shijie Wu, Haoran Li, Luke Zettlemoyer, and Veselin Stoyanov. 2020b. [Emerging cross-lingual structure in pretrained language models](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6022–6034, Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual Language Model Pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis,

- Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Federico Martelli, Najla Kalach, G. Tola, and Roberto Navigli. 2021. SemEval 2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Marco Maru, Federico Scozzafava, Federico Martelli, and Roberto Navigli. 2019. SyntagNet: Challenging supervised word sense disambiguation with lexical-semantic combinations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3534–3540, Hong Kong, China. Association for Computational Linguistics.
- George A Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.
- George A Miller, Claudia Leacock, Randee Teng, and Ross T Bunker. 1993. A semantic concordance. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop Held at Plainsboro, New Jersey, March 21-24, 1993*.
- Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69.
- Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020a. SensEmBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8758–8765.
- Bianca Scarlini, Tommaso Pasini, and Roberto Navigli. 2020b. With More Contexts Comes Better Performance: Contextualized Sense Embeddings for All-Round Word Sense Disambiguation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace’s Transformers: State-of-the-Art Natural Language Processing. *CoRR*, abs/1910.03771.

# FII\_CROSS at SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation

Ciprian Bodnar<sup>1</sup>, Andrada Tapuc<sup>1</sup>, Cosmin Pintilie<sup>1</sup>,  
Daniela Gifu<sup>1,2</sup>, Diana Trandabăț<sup>1,3</sup>

<sup>1</sup>Faculty of Computer Science, “Alexandru Ioan Cuza” University of Iasi

<sup>2</sup>Institute of Computer Science, Romanian Academy - Iasi Branch

<sup>3</sup>Imagination Play SRL

{ioan.bodnar, andrada.tapuc, cosmin.pintilie, daniela.gifu, dtrandabat}@info.uaic.ro

## Abstract

This paper presents a word-in-context disambiguation system. The task focuses on capturing the polysemous nature of words in a multilingual and cross-lingual setting, without considering a strict inventory of word meanings. The system applies Natural Language Processing algorithms on datasets from SemEval 2021 Task 2, being able to identify the meaning of words for the languages Arabic, Chinese, English, French and Russian, without making use of any additional mono- or multilingual resources.

## 1 Introduction

The computational task of disambiguating a word in the context of its sentence is still a very challenging topic facing natural language processing (NLP). In this study, we refer to word meaning that requires a multidisciplinary approach for its detection. From sense-based and contextualized embeddings, all tries are aimed at providing an understanding of words in context. We notice that evaluating such approaches is not easy. For instance, traditional Word Sense Disambiguation (WSD) fails to test latent representations unless these are linked to explicit sense inventories such as WordNet (Matuskevych, 2016) or BabelNet (Navigli and Pozetto, 2012; Luan *et al.*, 2020). To resolve the problem of disambiguation for both lingual dimensions, we tried to use a combination of well-known algorithms to provide an optimal system.

The legitimate research questions this paper intend to answer: *Is Word-in-Context Disambiguation a barrier for NLP techniques?*

The approach we propose in this paper investigates two models of cross-lingual word embeddings, comparing them to the shared-translation effect and the cross-lingual coactivation effects of false and true friends (cognates) found in human language. We find that the similarity structure of the cross-lingual word embeddings space yields the same effects as the human bilingual lexica (Merlo and Rodriguex, 2019). Research on bilingual lexica has uncovered fascinating interactions between the L1 (native language) and L2 (second language) lexica showing that both production and comprehension coactivate lexical items in both languages, indicating that bilinguals store lexical representations from their native and their second language in the same space.

The rest of the paper is organized as follows: section 2 describes the literature related to sense disambiguation, section 3 presents the dataset and method of this study, section 4 resumes the results of the conducted experiments, followed by section 5 with the conclusions and discussions about how to increase the accuracy.

## 2 Background

This topic has attracted significant attention in recent years, evidenced by increasing number of workshops (e.g., SemEval-2013 Task 10: Cross-lingual word Sense Disambiguation - CLWSD). Participating in such competitions is especially attractive since teams have thus access to labeled data.

For binary tasks, as the case of this competition, there are many computational models to be used in detecting the right word sense.

The recent advancements in corpus linguistics technologies, as well as the availability of more and more textual data, encourage many researchers to take advantage of comparable and parallel corpora to address different NLP tasks. Work on this topic is however highly subjective and biased. In general, the methods are based on Bag of Words features, usually normalized with  $tf*idf$  or character n-grams features for stylistic purposes.

Most approaches are supervised methods which can be classified into different methods:

(1) *regression*, based on the embeddings in one language using a leastsquares objective (Dinu et al., 2015; Artexe et al., 2018);

(2) *orthogonal*, based on the embeddings in one or both languages under the constraint of the transformation (Zhang et al., 2016; Smith et al., 2017);

(3) *canonical*, based on the embeddings in both languages to a shared space, using canonical link extension of it (Lu et al., 2015).

Also, several systems use an approach similar to ours' in considering Sent2vec the main algorithm (Pagliardini et al., 2018). Other systems used a *maxent* classifier trained over local context or even a KNN (*K-Nearest Neighbors*) classifier to solve the CLWSD (*Cross-Lingual Word Sense Disambiguation*) task. One of the interesting approaches was using machine translation (Baker et al., 1993). Although the winning systems for the CLWSD task used different approaches (statistical machine translation and classification algorithms), they also only used a parallel corpus to extract disambiguating information, while not using external resources such as WordNet. As a consequence, their system is very flexible and language-independent.

The topic of multilingual and cross-lingual disambiguation has attracted significant attention in recent years (Akyürek et al., 2020), with approaches ranging from learning effective vector representations (Loureiro and Jorge 2019, Scarlini et al., 2020) to infusing neural networks with knowledge graph information (Bevilacqua and Navigli 2020).

Our approach is more focused on recent research on the bilingual lexicon, which uncovered fascinating interactions between the lexica of the native language and that of the second language in bilingual speakers. Thus, it has been found that the lexicon of the underlying native language affects the organization of the second language (Riley et

al., 2020). In that spirit, our system includes distributed representations to disambiguate words in context.

### 3 Datasets and Methods

The dataset for the SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation is detailed in the task description paper (Martelli et al., 2021).

Language	Total words	Training words	Testing words
Arabic	20000	15000	5000
Chinese	16000	12000	4000
English	24500	16500	8000
French	22000	15500	6500
Russian	20000	14500	5500

Table 1: Corpus statistics

#### 3.1 Dataset

The dataset (Table 1), released in JSON format and divided into .data and .gold files, had the following composition: training, development and test subsets for multilingual and cross-lingual settings.

The data files contain the following information:

- (1) unique id of the pair;
- (2) target lemma;
- (3) part of speech;
- (4) first sentence;
- (5) second sentence;
- (6) start and end indices of the target word occurring in the first and second sentence.

The training data set contains 8000 entries for each multilingual language and 8000 entries for cross-lingual language combinations, and in the test dataset there are 1000 entries for each. The .gold files contain the following information, as exemplified below:

- unique id of the pair
- tag (binary, either T/F)

```
{
  "id": "training.en-en.0",
  "tag": "F"
},
```

We used NLTK to tokenize the sentences and removed the stop words, only keeping the lemmas in sentence1 and sentence2 respectively, and the



gold tag. After the transformation, the data in the training files look as exemplified below:

```
{
  "sentence1": "context
coordination integration Bolivia hold
key play process infrastructure
development ",
  "sentence2": "school water needed
girl sent fetch taking time away study
play ",
  "lemma": "play",
  "tag": "F"
},.
```

### 3.2 Methods

We considered that our system will first solve the multilingual part, followed by the cross-lingual part. Therefore, we propose the specific architecture presented in Figure 1.

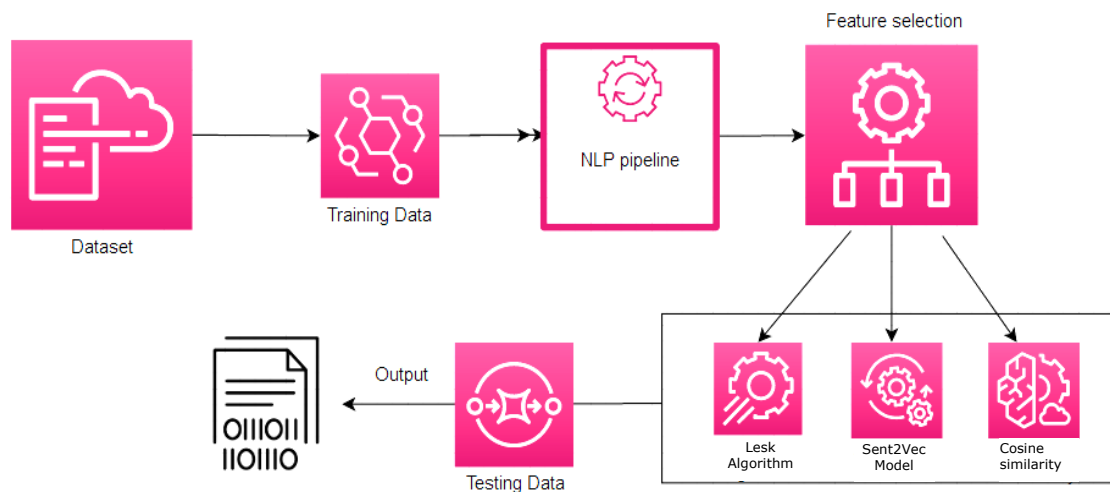


Figure 1. The System Architecture

Sent2Vec presents a simple but efficient unsupervised method to train distributed representations of sentences. It can be thought of as an extension of FastText and Word2vec (CBOW) to sentences. The sentence embedding is defined as the average of the source word embeddings of its constituent words. This model is furthermore augmented by learning source embeddings for both unigrams and various n-grams of words occurring in sentences and averaging the n-gram embeddings along with the words (Pagliardini et al., 2018). Thus, in our output, we have the initial tag from the labelled data, along with the score obtained through Sent2Vec.

As baseline, we used the Lesk algorithm. Thus, we extracted the definition of the queried target word from the first and the second sentence,

respectively. Finally, we compared the definitions of the target word in the two contexts, thus reaching the *True* or *False* tag. Lesk output before comparing definitions is presented below. Using NLTK-WORDNET, we extracted all synonyms for a target word.

Sentence 1: In that context of coordination and integration, Bolivia holds a key play in any process of infrastructure development.

```
-Sense:Synset('play.v.02')
-Definition:act or have an effect in
a specified way or with a specific
effect or outcome
```

Sentence 2: A musical play on the same subject was also staged in Kathmandu for three days.

```
-Sense:Synset('play.v.28')
-Definition:discharge or direct or
be discharged or directed as if in a
continuous stream
```

Our final approach was to combine the Lesk algorithm, along with Sent2Vec and vector cosine (Bojanowski et al., 2017). The cosine similarity is computed for each pair of sentences in our input. The pipeline used cross-lingually aligned versions of *fasttext* word vectors.

After running all modules, we obtained two scores, given by the cosine similarity and Sent2Vec

respectively, and a tag ( $T$  or  $F$ ) provided by the Lesk algorithm. In order to apply an integrative approach, we transformed the tags from the Lesk algorithm into a 3rd score: if the tag is  $T$  the Lesk score became 0.9, while if the tag is  $F$ , the score became 0.3. These values were determined empirically, after several tests with different weights. Below are the scores obtained for the pair of sentences discussed above.

```
sent2vec score: 0.0912621021270752
lesk score: 0.3
cosine score: 0.14142135623730948
gold tag: F
```

In order to establish a ranking on the three scores and their influence on the final tag, we tested and analyzed several combinations of weights, as follows:

```
Score1 - 30%sent2vec*10 + 30%lesk +
40%cosine*10: 0.9294717313304636
Score2 - 30%sent2vec*10 + 20%lesk +
50%cosine*10: 1.0408930875677729
Score3 - 30%sent2vec*10 + 40%lesk +
30%cosine*10: 0.818050375093154
Score4 - 40%sent2vec*10 + 20%lesk
+ 40%cosine*10: 0.9907338334575388
```

After a detailed error analysis in which we compared the gold tags from the development corpus with those issued in each of our combinations, we decided that the final tags be assigned according to the score brought by the formula (1):

```
(1) FiiCros_formula = 20%*Lesk + 30%
* Sent2Vec*10 + 50% Cosine *10
```

## 4 Results

The results for each individual task (Precision, Recall and F1-score) using the specific test dataset are presented: for multilingual test dataset (Table 2) and for crosslingual dataset (Table 3). The baseline identified 4483 correct tags out of 8000 inputs. After fine tuning the weights of the combination of our algorithms with the final formula discussed above our system reached 5760 correct tags out of 8000 inputs.

Model	Precision	Recall	F1-score
Lesk Baseline EN-EN	56%	65%	60,17%
Lesk + Sent2Vec + Cosine Vectorial EN-EN	72%	68%	69,94%
Lesk Baseline FR-FR	52%	61%	56,14%
Lesk + Sent2Vec + Cosine Vectorial FR-FR	70%	66%	67,94%
Lesk Baseline AR-AR	50%	59%	54,13%
Lesk + Sent2Vec + Cosine Vectorial AR-AR	68%	64%	65,94%
Lesk Baseline ZH-ZH	51%	60%	55,14%
Lesk + Sent2Vec + Cosine Vectorial ZH-ZH	69%	65%	66,94%
Lesk Baseline RU-RU	53%	62%	57,15%
Lesk + Sent2Vec + Cosine Vectorial RU-RU	71%	67%	68,94%

Table 2. Experimental Results for multilingual test dataset

Model	Precision	Recall	F1-score
Lesk Baseline EN-AR	53%	61%	56,71%
Lesk + Sent2Vec + Cosine Vectorial EN-AR	70%	66%	67,94%
Lesk Baseline EN-FR	49%	58%	53,12%
Lesk + Sent2Vec + Cosine Vectorial EN-FR	67%	64%	65,46%
Lesk Baseline EN-ZH	44%	55%	48,88%
Lesk + Sent2Vec + Cosine Vectorial EN-ZH	64%	60%	61,93%
Lesk Baseline EN-RU	47%	57%	51,51%
Lesk + Sent2Vec + Cosine Vectorial EN-RU	65%	61%	62,93%

Tables 3. Experimental Results for crosslingual dataset

We noticed to have lower scores when using the baseline (Lesk algorithm) than the combination of the three algorithms (Lesk, Sent2Vec and Cosine Vectorial).

Although it might have seemed to have similar results with the simple Word2Vec version, we considered Sent2Vec to be more reliable because it did not depend on the number of words in the vocabulary, and this was an advantage since the sizes of the vocabularies were diverse across languages.

## 5 Conclusion and Discussions

This paper presents a system participating at SemEval 2021 Task 2. Our solution indicates a good start for solving word sense disambiguation.

The main challenge behind word sense disambiguation is to make ample use of the available technologies since ambiguities in any language provide great difficulty in the use of information technology. The major difficulty lays in the fact that words in human language can be interpreted in more than one way, depending on the context (Tan, 2013).

Since we performed a detailed investigation of monolingual and bilingual disambiguation, our experimental results showed that Sent2vec and Lesk approaches are remarkably efficient for both tasks. The overall results are satisfactory and exceed the baseline; however, there is still room for improvement.

A larger and well-annotated dataset would provide more opportunities for exploring the issue of disambiguation. Additionally, building a dataset sufficient in size and diversity will allow experiments with deep learning methods. The biggest challenge in this project was working in different languages at the same time while some tools were available to English only.

From our research, we noticed that the average scores are around 75% when applying separately Lesk and Sent2Vec (or even Word2Vec), and it seems to be similar when combining the two.

## References

- Akyürek, A. F., Guo, L., Elanwar, R., Ishwar, P., Betke, M., Wijaya, D. T. (2020). *Multi-Label and Multilingual News Framing Analysis*. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 8614-8624).
- Artetxe, M., Labaka, G. and Agirre, E. (2018). *A robust self-learning method for fully unsupervised cross-lingual mappings of word embeddings*. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 789-798.
- Bevilacqua, M., & Navigli, R. (2020). *Breaking through the 80% glass ceiling: Raising the state of the art in Word Sense Disambiguation by incorporating knowledge graph information*. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 2854-2864).
- Baker, M., Francis, G., and Tognini-Bonelli, E. (1993). *Corpus Linguistics and Translation Studies: Implications and Applications*, Chapter 2. John Benjamins Publishing Company, Netherlands.
- Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). *Enriching Word Vectors with Subword Information*. Transactions of the Association for Computational Linguistics.
- Dinu, D., Lazaridou, A., and Baroni, M. (2015). *Improving zero-shot learning by mitigating the hubness problem*. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), workshop track.
- Lefever, E., Hoste, V. (2013). *SemEval-2013 Task 10: Cross-lingual Word Sense Disambiguation*. In Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the 7<sup>th</sup> SemEval, pp. 158-166.
- Daniel Loureiro and Al'ipio Jorge (2019). *Language modelling makes sense: Propagating representations through WordNet for full-coverage word sense disambiguation*. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5682–5691.
- Luan, Y., Hauer, B., Mou, L., Kondrak, G. (2020). *Improving Word Sense Disambiguation with Translations*. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 4055-4065.
- Lu, A., Wang, W., Bansal, M., Gimpel, K., and Livescu, K. (2015). *Deep Multilingual Correlation for Improved Word Embeddings*. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 250-256.
- Martelli F., Kalach N., Tola G., Navigli R. (2021) *SemEval-2021 Task 2: Multilingual and Cross-lingual Word-in-Context Disambiguation (MCL-WiC)*. In Proceedings of the 15th Workshop on Semantic Evaluation 2021.

- Matusevych, Y. (2016). *Learning Constructions from Bilingual Exposure. Computational Studies of Argument Structure Acquisition*. PhD. Thesis, Tilburg University.
- Merlo, P., Rodriguex, M. A. (2019). *Cross-lingual Word Embeddings and the Structure of the Human Bilingual Lexicon*. In Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), pp. 110-120.
- Navigli, R. and Ponzetto, S. P. (2012). *Joining Forces Pays off: Multilingual Joint Word Sense Disambiguation*. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1399-1410.
- Pagliardini Matteo, Gupta Prakhar, Jaggi Martin (2018) *Unsupervised Learning of Sentence Embeddings Using Compositional n-Gram Features*. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)
- Riley, P., Caswell, I., Freiag, M., Grangier, D. (2020). *Translationese as a Language in “Multilingual” NMT*. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7737-7746.
- Scarlina Bianca, Tommaso Pasini, and Roberto Navigli (2020) *SensEmBERT: Context-enhanced sense embeddings for multilingual word sense disambiguation*. In Proceedings of the AAAI Conference on Artificial Intelligence.
- Smith, S., L., Turban, D. H. P., Hamblin, S., and Hammerla, N. Y. (2017). *Offline Bilingual Word Vectors, Orthogonal Transformations and the Inverted Softmax*. In Proceedings of 5th ICLR.
- Tan, I. (2013). *Examining Crosslingual Word Sense Disambiguation*. MSc Thesis, Nanyang Technological University - <http://compling.hss.ntu.edu.sg/pdf/2013-masters-tan-liling.pdf>
- Zhang, Y., Gaddy, D., Barzilay, R., and Jaakkola, T. (2016). *Ten pairs to tag – multilingual pos tagging via coarse mapping between embeddings*. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1307-1317.

# XRJL-HKUST at SemEval-2021 Task 4: WordNet-Enhanced Dual Multi-head Co-Attention for Reading Comprehension of Abstract Meaning

Yuxin Jiang\*, Ziyi Shou\*, Qijun Wang, Hao Wu, Fangzhen Lin

HKUST-Xiao Robot Joint Lab on Machine Learning and Cognitive Reasoning

Department of Computer Science and Engineering

The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

{yjiangcm, zshou, qwanged, hwubx}@connect.ust.hk, flin@cse.ust.hk

## Abstract

This paper presents our submitted system to SemEval 2021 Task 4: *Reading Comprehension of Abstract Meaning*. Our system uses a large pre-trained language model as the encoder and an additional dual multi-head co-attention layer to strengthen the relationship between passages and question-answer pairs, following the current state-of-the-art model DUMA. The main difference is that we stack the passage-question and question-passage attention modules instead of calculating parallelly to simulate re-considering process. We also add a layer normalization module to improve the performance of our model. Furthermore, to incorporate our known knowledge about abstract concepts, we retrieve the definitions of candidate answers from WordNet and feed them to the model as extra inputs. Our system, called WordNet-enhanced DUAL Multi-head Co-Attention (WN-DUMA), achieves 86.67% and 89.99% accuracy on the official blind test set of subtask 1 and subtask 2 respectively.

## 1 Introduction

Recently, there has been an increasing interest on Machine Reading Comprehension (MRC). While most MRC studies such as CNN/Daily Mail (Hermann et al., 2015) focus on concrete concepts, SemEval 2021 Task 4, *Reading Comprehension on Abstract Meaning* (ReCAM), targets abstract concept understanding, including *imperceptibility* in subtask 1 and *nonspecificity* in subtask 2. The former, *imperceptibility*, highlights the abstract words that refer to ideas and concepts that do not correspond directly to human perception. The latter is for hypernyms and abstract concepts such as the class of vertebrate which includes whales as a concrete subclass (Changizi, 2008).

\*Equal contribution.

Passage

The 29-year-old Belgium international, whose old deal ran until 2018, has made 179 appearances for the Premier League club since his 2012 move from Ajax. "It's a big relief. The future looks great so I'm very happy to be a part of it," he said. "This is an unbelievable group of talent. There's a great buzz around Tottenham." Vertonghen's new deal comes a day after striker Harry Kane signed a contract until 2022.

Question

Tottenham Hotspur defender Jan Vertonghen has signed a new contract, @placeholder him to the club until 2019.

Candidate Answers

dedicated

connecting

enabling

prompting

committing

Figure 1: An example of ReCAM subtask 1.

In this task, given news fragments and incomplete abstracts, the machine needs to select the most suitable abstract words from candidate answers. Figure 1 shows one example of ReCAM subtask 1. Passage is the news sections and Question is a human written summary in which abstract words have been removed. Machines are requested to choose abstract words from five candidates for replacing @placeholder.

For this shared task, we regard both subtasks as multi-choice MRC tasks. Various deep neural networks and attention mechanisms (e.g. (Dhingra et al., 2017; Wang et al., 2018; Zhang et al., 2020a,b; Jin et al., 2020)) have been proposed to address these tasks. In our work, following the state-of-the-art model DUMA (Zhu et al., 2020), we adopt a Pre-trained Language Model (PrLM) as encoder and extend with an additional dual multi-head co-attention layer to strengthen the relationship between passages and question-answer pairs. For the dual multi-head attention layer, while DUMA builds passage-question and question-passage attention modules in a parallel way to simulate the transposition thinking process,

our model stacks two attention modules in order to simulate the process of re-considering for a deeper understanding of the passage. More details on our attention calculation process can be found in Section 2. Furthermore, we add an additional layer normalization module immediately after the attention module. From our experiments, we found that this additional normalization module definitely improves our model’s performance.

Our most significant design decision is to use WordNet (Miller, 1995) to expand on the abstract concepts in the candidate answers. Intuitively, expanding an abstract concept according to its definition in a dictionary should help as it helps relate the abstract concept with others that may occur in the text. A key conclusion that we can draw from our experiments is that this is indeed the case. One problem that we encountered when implementing this idea was that most English words have multiple entries in WordNet. For example, *bank* in WordNet can have Noun definitions as well as Verb definitions. We addressed this problem by using some heuristics and some additional information such as part-of-speech labels. Because of the significant role played by WordNet, we call our system **WordNet-enhanced DUal Multi-head Co-Attention (WN-DUMA)**.

We remark that our system did not use any additional training data for the tasks. In the final evaluation, our model is ranked 10 out of 23 and 9 out of 28 on the official subtask 1 and subtask 2 blind test set with 86.67% and 89.99% accuracy, respectively. The code for our model is publicly available<sup>1</sup>.

The rest of the paper is organized as follows. Section 2 gives the details of our system. Section 3 describes our experimental setup including the datasets and hyper parameters used for training. Section 4 presents experimental results. Section 5 concludes this paper with some final remarks.

## 2 System Description

In this section, We describe the framework of our end-to-end model WN-DUMA. Figure 2 depicts the detailed architecture of our approach, with inputs at the bottom and outputs at the top.

**WordNet-enhanced Encoder** We regard both subtask 1 and 2 as multi-choice MRC problems. Such a problem includes a passage, a question with

<sup>1</sup><https://github.com/zzshou/RCAM>

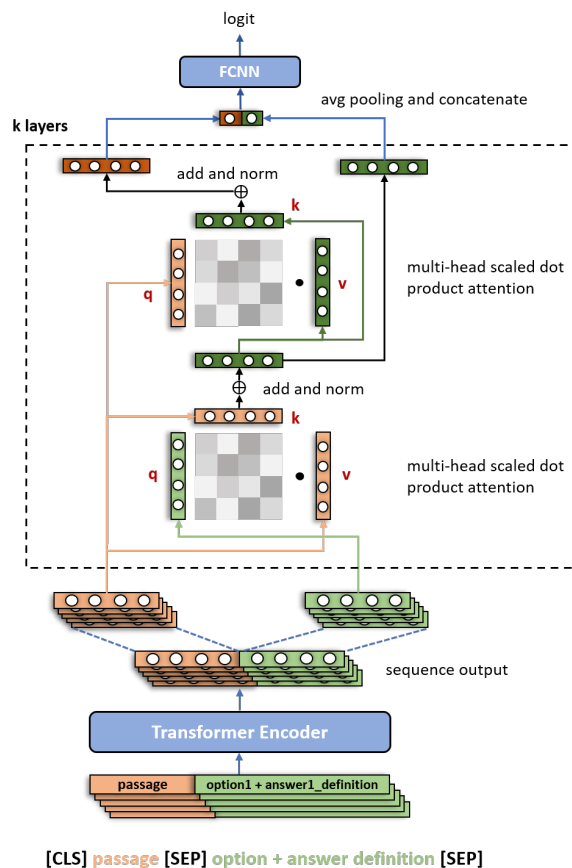


Figure 2: The overall model architecture.

a *@placeholder*, and 5 candidate answers to choose from. First, we replace *@placeholder* in the question with the given 5 candidate answers to form 5 options. In the tasks, the candidate answers are all single words with abstract meanings, so we decided to add some extra knowledge from WordNet (Miller, 1995) to help the system better understanding the abstract meanings. More specifically, for a single candidate answer, we find its part-of-speech tag based on the option it’s located in, and extract its definitions under this part-of-speech tag. After tokenization, every instance is cast into the input form: [CLS] *passage* [SEP] *option + answer definition* [SEP]. To encode input tokens into representations, we feed them through a PrLM based on Transformer to obtain sequence embeddings, which draws a global relationship between the passage and the option-definition.

**Dual Multi-head Co-Attention Layer** Based on the above process, we further separate the output representations from transformer encoder to acquire the passage context embeddings  $E^P \in \mathbb{R}^{d_{model} \times l_p}$  and the context embeddings of option-

definition  $E^{OD} \in \mathbb{R}^{d_{model} \times l_{od}}$ , where  $l_p, l_{od}$  denote the maximum length of passage and option-definition respectively. Then based on the bi-directional matching network of DUMA which is quite similar to the multi-head self-attention module in vanilla transformer block (Vaswani et al., 2017), we first take  $E^{OD}$  as Query,  $E^P$  as Key and Value to calculate one of the co-attention representations, which simulates the process of human re-reading the passage with impression of option and definition. The formulas are listed as follows:

$$Q_i = E^{OD}W_i^Q \quad (1)$$

$$K_i = E^P W_i^K \quad (2)$$

$$V_i = E^P W_i^V \quad (3)$$

$$head_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}}) V_i \quad (4)$$

$$MHA = Concat(head_1, \dots, head_h) W^O \quad (5)$$

$$REP_1 = Normalize(E^{OD} + MHA) \quad (6)$$

where  $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$ ,  $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ ,  $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ ,  $W_i^O \in \mathbb{R}^{h d_v \times d_{model}}$  are linear transformations with learnable parameters,  $d_q, d_k, d_v$  denote the dimension of *Query*, *Key* and *Value*,  $h$  denotes the number of heads. Different from the structure of DUMA, here we make two changes: 1) apply "Add and Normalize" after getting the multi-head attention representation, which could result in more stable training. 2) compute another co-attention representation by stacking rather than paralleling: take the acquired  $REP_1$  as *Key* and *Value*,  $E^P$  as *Query*, which simulates the process of re-considering the option-definition with deeper understanding of the passage. Finally, we obtain  $REP_1$  and  $REP_2$ , which have the same dimension as  $E^{OD}$  and  $E^P$ , respectively. As a result, we can stack the co-attention module for  $k$  layers.

**Classifier** Here the co-attention representations  $REP_1$  and  $REP_2$  are merged and used for final classification:

$$I_1 = AvgPool(REP_1) \quad (7)$$

$$I_2 = AvgPool(REP_2) \quad (8)$$

$$M = Concat(I_1, I_2) \quad (9)$$

$$logits = MW_M \quad (10)$$

where  $I_1, I_2 \in \mathbb{R}^{d_{model}}$ ,  $M \in \mathbb{R}^{2d_{model}}$ ,  $W_M \in \mathbb{R}^{d_{model} \times n_{class}}$  denotes the one-layer fully-connected neural network,  $n_{class}$  denotes the number of candidate answers. Consequently, for a single instance, we could get as many logits as the

	Task 1	Task 2
Train	3,227	3,318
Dev	837	851
Test	2,025	2,017
Avg. passage length	270.3	429.7
Avg. question length	24.6	27.1
Vocabulary size	16,318	17,006
Answer vocabulary size	4,333	4,775

Table 1: Basic statistics of subtask 1 and subtask 2 dataset.

candidate answers, which are used to compute the cross-entropy loss by softmax.

### 3 Experimental Setup

**Data and Metric** We used the official datasets (Zheng et al., 2021) provided by SemEval 2021 Task 4 competition. They were collected from BBC News in English language. Some basic statistics are listed in Table 1. According to the requirement of the organizers, participants could only use the corresponding dataset for a specific subtask to build models to ensure fairness. For better performance, technics like multi-task learning (Wan, 2020) are recommended for MRC tasks. In both subtask 1 and subtask 2, we utilize accuracy as the metric to evaluate our model performance.

**Hyper Parameters** All of our codes are written based on PyTorch<sup>2</sup>. To extract the word definition of candidate answers, we use NLTK toolkit (Bird et al., 2009). The transformer encoder we used is pretrained ALBERT-xxlarge-v2 model<sup>3</sup>. Since the code of DUMA is not open-source, we reimplement it by only using one co-attention layer where the attention heads are 64 and the dimension of *Query*, *Key* and *Value* are all 64, because it is pointed that more co-attention layers do not improve the performance (Zhu et al., 2020). The setting is also applied to our WN-DUMA for fair comparison.

Due to limited resources, the maximum sequence length of input tokens is set to 150 for both subtask 1 and subtask 2. In fact, we found that sequence length longer than 150 can only slightly improve the model performance. We choose mini-batch size equal to 2, and the AdamW optimizer

<sup>2</sup><https://pytorch.org/>

<sup>3</sup><https://github.com/huggingface/transformers>

(Loshchilov and Hutter, 2018) with an initial learning rate of  $5e-06$ . We use some strategies for more stable training: 1) clip the gradient norm to 10; 2) adopt a linear scheduler with warm up of the first 10% training steps. To avoid overfitting, we apply 0.1 dropout (Srivastava et al., 2014) rate to the co-attention layer. We trained all the models for 3 epochs, evaluate on the dev set at every 200 training steps and save the model with the best dev accuracy. For each single model, we run experiments for 5 times with different random seeds and use the average as the ultimate performance.

## 4 Results

### 4.1 Quantitative Analysis

Table 2 summarizes the experimental results. The first three models only have encoder (without enhancement of WordNet) and classifier part. It is clearly seen that ALBERT is much more efficient as encoder for abstract meaning understanding. It is worth noting that by only using the question and answer as input, the ALBERT model can also get pretty good results, as table 2 shows. Intuitively, it may be because the model could utilize syntax and semantics of the question sentence to choose the correct answer without looking through the passage.

Compared to ALBERT<sub>xxlarge</sub>, adding DUMA layer obtains around 0.2% improvement in subtask 1, and more than 3% improvement in subtask 2. Besides, our WN-DUMA single model achieves further improvements based on DUMA on both subtasks, +0.83% and +1.3% respectively, without increasing the number of parameters. Using a majority vote scheme, we ensemble our WN-DUMA model with different parameters for more stable predictions. Eventually, our ensemble models which get 87.57% on subtask 1 dev set and 90.01% on subtask 2 dev set acquire the best performance on test sets (86.67% and 89.99%, respectively) among our submissions.

Figure 3 and Figure 4 illustrate the dev accuracy of different models on subtask 1 and subtask 2 as the number of training steps increases. It is interesting to observe that models with co-attention layer (DUMA and WN-DUMA) could get over 70% accuracy with only 10% of training examples. While ALBERT model has to be trained with the full dataset to get relatively high accuracy. Consequently, our WN-DUMA model may be useful when there only exists a small amount of training

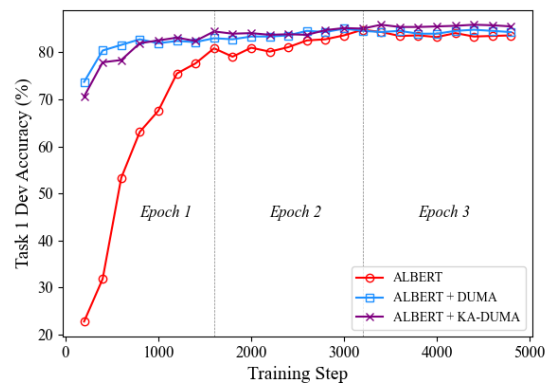


Figure 3: Subtask 1 dev accuracy over number of training steps.

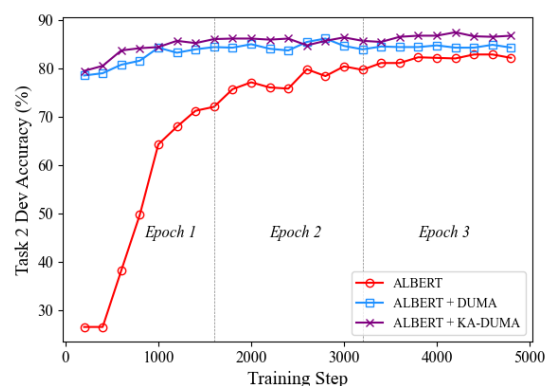


Figure 4: Subtask 2 dev accuracy over number of training steps.

data.

### 4.2 Error Analysis

In order to further improve our model performance in the future, we analyze some incorrect predictions made by WN-DUMA, and classify them into two categories:

- Candidate answers with similar meanings. In some failure cases, the similarities between candidate answers are too high to distinguish. For example, outstanding and extraordinary, challenge and attempt, etc.
- Lack of commonsense and relying too much on the information of the passage. Due to the fact that the question is the summary of the passage, the machine need to choose the most appropriate answer from a global perspective with some commonsense. However, our model make decisions by only capturing the local information in some cases. A spe-



Model	Task 1 dev	Task 1 test	Task 2 dev	Task 2 test
BERT <sub>large</sub> (Devlin et al., 2019)	67.74	-	69.45	-
RoBERTa <sub>large</sub> (Liu et al., 2019)	74.31	-	74.50	-
ALBERT <sub>xxlarge</sub> (Lan et al., 2020)	84.83	-	82.84	-
ALBERT <sub>xxlarge</sub> + DUMA (Zhu et al., 2020)	85.07	-	86.13	-
ALBERT <sub>xxlarge</sub> (question only)	79.57	-	82.14	-
ALBERT <sub>xxlarge</sub> + WN-DUMA (single)	<b>85.90</b>	84.54	<b>87.43</b>	86.61
ALBERT <sub>xxlarge</sub> + WN-DUMA (ensemble)	<b>87.57</b>	86.67	<b>90.01</b>	89.99

Table 2: Model comparison on subtask 1 and subtask 2 dataset.

Passage

Joe Wincott of The Sandon School in Chelmsford, Essex said an extra £1.3bn promised by the government was too late to help it in the next academic year. The head teacher said cutting lessons from 26 to 25 hours a week would allow him to balance the school budget. Education Secretary Justine Greening said per pupil funding was set to go up from £4,100 to £4,800 in 2018. Mr Wincott said the school budget had been cut by £450,000 since 2011 and he had reduced the costs of "everything from power supplies, examination and photocopying". He said: "We were down to the situation where we were unable to balance our budget for 2017-2018, so we took the decision... to drop to 25 hours, which is what most schools deliver." Parents at the mixed comprehensive school, which has 1,270 pupils aged 11 to 18 and was rated good in its last Ofsted inspection, had been "remarkably understanding". But the head teacher added a "significant number of pupils" were entering the school system, and pay, pension and National Insurance contributions were all due to increase. As a result, he believes the promised extra pupil funding in 2018 will "probably put us where we are now, but without having to make cuts to staff".

Question

A @placeholder school will cut an hour of teaching a week from the autumn in a bid to save £ 100,000.

Candidate Answers

secondary    temporary    troubled    new    third

✓    predicted

Figure 5: An failure example made by WN-DUMA. The ground true is the answer with a correct mark at the bottom. While the prediction is the answer with "predicted" at its bottom.

cific example can be seen in Figure 5. We can see that the model predicts that the answer is "troubled", most likely because the passage mentions "the school was trapped into financial difficulties".

## 5 Conclusion

In this paper, we describe our submitted system in SemEval 2021 Task 4 ReCAM. Unlike previous MRC datasets, ReCAM focus more on machine's ability in understanding and representing abstract concepts. In order to provide more knowledge of abstract word, we extract WordNet definitions for each candidate answer based on part-of-speech tags. In addition, our proposed WN-DUMA model consists of a PrLM as the encoder and a dual multi-head co-attention layer to enhance the relationship

between passage and question-answer pairs as human's re-considering process. Our WN-DUMA model improves the performance of our baseline model DUMA on these datasets.

There are some limitations in our experiments. Firstly, training data size of this task is limited compared to other MRC tasks, with less than 3400 training pairs in both subtasks. This is understandable as collecting labeled data in many natural language processing tasks is expensive. Secondly, using ALBERT<sub>xxlarge</sub> PrLM, we only set 150 as the maximum text length in our experiments due to device limitation. Important sentences in the passage that are highly relevant to the summary are sometimes not covered. For PrLMs, their performance always improve as the number of their parameters increase. The use of large pre-trained models sometimes requires the sacrifice of context. For our future work, we plan to explore ways to train models more efficiently with limited amount of labeled data, and to design more cost-effective models to deal with long input texts.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.
- Mark A Changizi. 2008. Economically organized hierarchies in wordnet and the oxford english dictionary. *Cognitive Systems Research*, 9(3):214–228.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov.

2017. [Gated-attention readers for text comprehension](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846. Association for Computational Linguistics.
- Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1693–1701.
- Di Jin, Shuyang Gao, Jiun-Yu Kao, Tagyoung Chung, and Dilek Hakkani-tur. 2020. [Mmm: Multi-stage multi-task learning for multi-choice reading comprehension](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8010–8017.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A lite BERT for self-supervised learning of language representations](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2018. Fixing weight decay regularization in adam.
- George A. Miller. 1995. [Wordnet: A lexical database for english](#). *Commun. ACM*, 38(11):39–41.
- Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. [Dropout: a simple way to prevent neural networks from overfitting](#). *J. Mach. Learn. Res.*, 15(1):1929–1958.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.
- Hui Wan. 2020. Multi-task learning with multi-head attention for multi-choice reading comprehension. *arXiv preprint arXiv:2003.04992*.
- Shuohang Wang, Mo Yu, Jing Jiang, and Shiyu Chang. 2018. [A co-matching model for multi-choice reading comprehension](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 746–751. Association for Computational Linguistics.
- Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2020a. [Dcmn+: Dual co-matching network for multi-choice reading comprehension](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9563–9570.
- Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2020b. [Sgnet: Syntax-guided machine reading comprehension](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9636–9643.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Duma: Reading comprehension with transposition thinking. *arXiv preprint arXiv:2001.09415*.

# UoR at SemEval-2021 Task 4: Using Pre-trained BERT Token Embeddings for Question Answering of Abstract Meaning

Thanet Markchom, Huizhi Liang

University of Reading  
White Knights, Berkshire, RG6 6AH  
United Kingdom

t.markchom@pgr.reading.ac.uk, huizhi.liang@reading.ac.uk

## Abstract

Most question answering tasks usually focus on predicting concrete answers, e.g., named entities. These tasks can be normally achieved by understanding the contexts without additional information required. In Reading Comprehension of Abstract Meaning (ReCAM) task, the abstract answers are introduced. To understand abstract meanings in the context, additional knowledge is essential. In this paper, we propose an approach that leverages the pre-trained BERT Token embeddings as a prior knowledge resource. According to the results, our approach using the pre-trained BERT outperformed the baselines. It shows that the pre-trained BERT token embeddings can be used as additional knowledge for understanding abstract meanings in question answering.

## 1 Introduction

Question answering (QA) is one of the machine reading comprehension tasks. The goal is to find an answer of a given question based on a given context. In most QA tasks (Chen et al., 2016; Lai et al., 2017), the answers are commonly concrete words appearing in the contexts. Abstract words, on the other hand, have usually been ignored in such tasks. Unlike concrete words, these abstract words are difficult to understand since they cannot be perceived directly with human senses. To study machine comprehension of abstract meaning, a task called Reading Comprehension of Abstract Meaning (ReCAM) (Zheng et al., 2021) was proposed. Unlike other QA tasks, the answers in ReCAM are abstract words that used to summarise the information in the contexts.

ReCAM is divided into three subtasks. For subtask 1, the abstract answers are in the form of imperceptible words such as ‘objective’, ‘chance’, and ‘prospective’. Subtask 2 is about nonspecificity. The answers in this subtask are abstract words that

represent nonspecific or general concepts such as ‘vertebrate’. Subtask 3 focuses on generality of the models developed in the first two subtasks. In this subtask, the model in subtask 1 must be evaluated on subtask 2 data and vice versa. Based on these abstract answers, a machine has to understand the abstract meaning of certain words. This requires an additional knowledge to fulfill the lack of abstract concept information (Bi et al., 2019). For example, given a context which is a passage and a question shown in Table 1. In the example of Table 1, the context contains a passage and a question. The answer ‘neglected’ does not appear anywhere in the given passage. However, the sentence, ‘it gradually fell into disrepair in the late 1980s’, provides a clue to answer the question. If the model has a prior knowledge that the word ‘disrepair’ have a connection with ‘neglect’, then the correct answer will be chosen.

In this work, we propose to use a pre-trained word/token embedding model to provide external knowledge for abstract meaning understanding. The pre-trained BERT token embedding is chosen in this work due to its good performance in various natural language processing tasks. In our approach, the token embeddings are firstly extracted by the pre-trained BERT model. Presumably, these embeddings are enriched with additional information for understanding the abstract meanings. These embeddings are used as inputs in our approach to predict answers. In this way, the prior knowledge from the pre-trained model can be leveraged in the task of abstract answer prediction.

## 2 Related Work

A QA task usually requires a machine to understand a context to answer a question. Typically, a context is a passage and an answer usually appear somewhere in a given passage. Several ap-

proaches have been proposed to compute a combined passage-question representation and then use it to find an answer. The DeepLSTM Reader (Hermann et al., 2015) uses a deep long-term short-term memory (LSTM) encoder to find a representation of a concatenated passage-question text. The Attentive Reader (Hermann et al., 2015) attentively aggregates words to compute a passage representation based on a question. The Attentive Reader was modified in (Chen et al., 2016) where a deep LSTM module with the dot product attention function were replaced by a shallow bi-directional LSTM module with the bi-linear attention function. Recently, Dhingra et al. (2017) proposed a model called Gated-Attention (GA) Reader. It combines a multi-hop architecture and a novel attention mechanism to learn the representations. These models have shown good performances in question answering when answers are concrete words. However, in ReCAM task, the answers are abstract. Only the given contexts may not suffice to answer the questions.

Recently, word/token embeddings from pre-trained models have been popularly used in many applications. They are capable of providing additional knowledge from the resources they were pre-trained. Several pre-trained models have been proposed in the past few years such as GloVe (Pennington et al., 2014) and ELMo (Peters et al., 2018). Many models have also been developed from the transformer architecture (Vaswani et al., 2017), e.g., BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and DistilBERT (Sanh et al., 2020). They have been highly successful in pre-training token embeddings for various downstream tasks including QA. However, there has been no work where such models are applied in a QA task that focuses on imperceptibility and nonspecificity abstractness.

### 3 Using pre-trained token embeddings for a QA task

Each sample in the dataset of both subtasks consists of three components: passage, question and options. Each passage is a long text that is used to provide context for answering a question. Each question is a passage-summarized text containing one special token, *@placeholder*. This token is a representative of a word (an answer) that should be filled to complete the text. The correct answer must be selected from five possible options provided in each sample. Table 1 shows one exam-

ple in subtask 1. In the table, the first row is the passage providing a context for this sample. The second row is the question, which is a sentence that summarises the given passage. It contains one placeholder indicated by *@placeholder* token. This placeholder should be replaced by an abstract word corresponding to the passage. The third row is the list of options that can be selected to replace the placeholder in the question. The correct options is marked in bold which is 'neglected' in this case. The last row is the label indicates the answer of this example ranging from 0 (the first option) to 4 (the last option).

Given a context (a passage and a question containing a placeholder token), a placeholder embedding extracted from the pre-trained BERT model should be able to guide an answer. Due to the fact that the BERT model considers token contexts to learn token embeddings, a placeholder embedding should comprise information of its context. That means any word or token with a similar embedding should also be in the same context as a placeholder as well. Thus, to find the correct answer from a list of options, every option embedding extracted from the pre-trained BERT model is compared with a placeholder embedding. Then, any option with the most similar embedding compared to a placeholder embedding should be an answer. Accordingly, we propose an approach that considers similarity between placeholder and option embeddings to predict answers. Several metrics such as dot product, cosine similarity and euclidean distance can be considered to measure the similarity. The selected metric in this work is described in Section 4.

To extract placeholder and option embeddings in each sample, a question and a passage are firstly concatenated in the following form:

[CLS] + Question + [SEP] + Passage + [SEP]

This form is conventional in BERT framework. Normally, a '[CLS]' token is added for a classification purpose. It is usually represented a sentence level embedding. However, it is not utilized in our approach since we utilized embeddings in a token level. Two '[SEP]' tokens in the form are used to separate a question and a passage. Similarly, for every option in each sample, both '[CLS]' and '[SEP]' tokens are added as follows:

[CLS] + Option + [SEP]

Then, the pre-trained BERT model is used to extract embeddings from these prepared inputs.

<b>Passage</b>	The Trainspotting author has agreed to become patron of The Leith Theatre and launch a new fundraising drive. The Leith Theatre Trust took over the lease of the art deco venue from City of Edinburgh Council last year ... However, it gradually fell into disrepair in the late 1980s and eventually had to be closed down by the council ...
<b>Question</b>	Irvine Welsh is to spearhead a campaign aimed at reviving a @placeholder theatre in Leith 30 years after its last show.
<b>Options</b>	(A) <b>neglected</b> , (B) renewed, (C) lavish, (D) revised, (E) proposed
<b>Label</b>	0

Table 1: An example of a passage, a question, and five options. The task is to select the correct answer (bold) for replacing @placeholder

The placeholder embedding is extracted from the question-passage concatenated input and each option embedding is extracted from each option input. These embeddings are used as inputs of the proposed approach. As for targets, all labels given are converted to one-hot vectors. With the placeholder and options embeddings as inputs and the one-hot vectors as targets, our proposed approach learn how to predict the probabilities that each option is an answer for each sample. Figure 1 shows the input and target generated from an example shown in Table 1. The pre-trained BERT model is used to extract embeddings of the placeholder and option tokens in the prepared format. The embeddings are the hidden weights of the given tokens from the last hidden layer of the pre-trained BERT model. The label 0 is converted to a one-hot vector  $[1, 0, 0, 0, 0]$ . These embeddings are then passed through the prediction model. This prediction model consists of six learning modules for learning placeholder and five option embeddings in each sample. These modules consist of a dense layer with an output size 768 and a tanh activation function. They take a 768-dimensional pre-trained BERT token embedding and outputs the fine-tuned embedding with the same size. For each placeholder embedding, the fine-tuned embedding is produced by the placeholder embedding module. Similarly, for each option embedding, the fine-tuned option embedding is also obtained from the corresponding option learning module. However, using separate option learning module for each option may cause a bias in selecting some options at the end. We therefore propose to use shared-weight modules for learning all fine-tuned option embeddings simultaneously. In other words, all options in each sample are learned by the modules that share the same weights. After that, for

each option, the similarity between the fine-tuned placeholder and the fine-tuned option embedding is computed. All of the similarities from all options are then concatenated to form a vector with the size 5. A softmax activation function is applied on the concatenated vector to produce the final output with the same size. This output vector represent the probabilities that each option is an answer of a given sample. The entire proposed approach is illustrated in Figure 2. In the figure,  $p$  denotes the fine-tuned placeholder embedding while  $o_0, o_1$  and  $o_4$  denote the first, the second and the last option embedding respectively.  $s$  denotes a similarity function.  $s(p, o_i)$  is the similarity between  $p$  and any  $o_i$ .

## 4 Experimental setup

The pre-trained BERT model used in this work is BERT-Base-Uncased<sup>1</sup> (Turc et al., 2019). All inputs were converted to lower case and tokenized by BERT-Uncased tokenizer. The prediction model was trained using RMSprop optimizer for 100 epochs with the learning rate set to 0.001. A categorical cross entropy was used as a loss function. To avoid over-fitting, both L1 and L2 regularizers were added at the dense layer in both placeholder and option learning modules. The regularization factors were set to 0.001. To select the similarity metric, we examined three functions, i.e., dot product ( $s_d$ ), cosine similarity ( $s_c$ ) and euclidean-distance-based similarity ( $s_e$ ) computed by

$$s_d(p, o) = p \cdot o \quad (1)$$

$$s_c(p, o) = \frac{p \cdot o}{\|p\| \|o\|} \quad (2)$$

$$s_e(p, o) = \frac{1}{1 + \|p - o\|} \quad (3)$$

<sup>1</sup><https://github.com/google-research/bert>

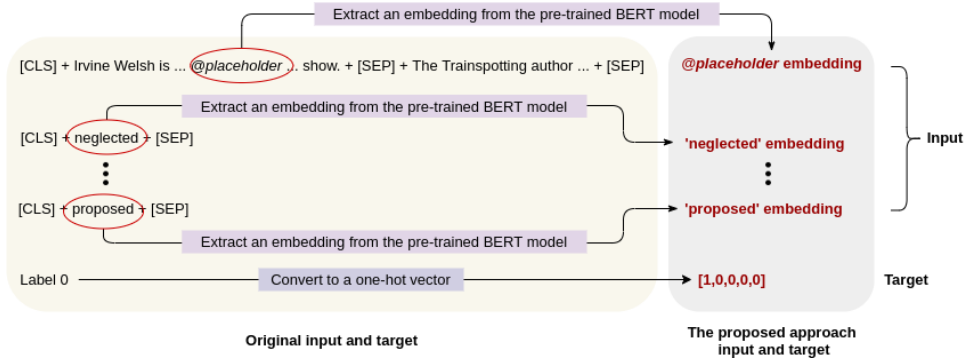


Figure 1: The input and target generated from an example shown in Table 1

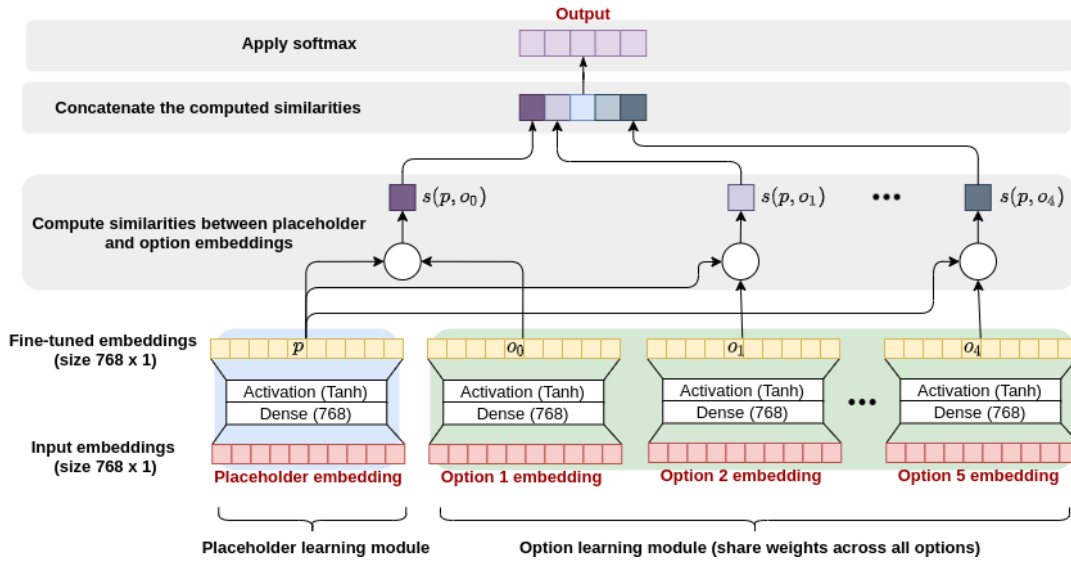


Figure 2: The proposed approach framework

where  $p$  denotes a placeholder embedding and  $o$  denotes an option embedding. We applied these metrics in the proposed approach and performed experiments on the development sets provided. The results are shown in Table 2. As in the table, the proposed approach using a dot product performed best compared to the others. Hence, a dot product was used in the proposed approach when it was applied on the trial and test sets

Similarity metric	Accuracy	
	Subtask 1	Subtask 2
$s_d$	0.48	0.46
$s_c$	0.34	0.31
$s_e$	0.33	0.31

Table 2: Evaluation results of the proposed approach using different similarity metrics on the development sets

#### 4.1 baselines

We compared the proposed approach named as **BERT-S** with other three baselines as follows:

- **MLP**: a multilayer perceptron (MLP) binary classifier. It consists of two hidden layers with the size 512 and 128 with a rectified linear unit (ReLU) activation function. The output is produced by an output layer with the size 1 and a sigmoid activation function. Each input is a concatenated placeholder-option embedding and its label is 1 if the option is an answer and 0 otherwise. For every sample, an answer is chosen from the concatenated placeholder-option embedding that gives the highest prediction value. A binary cross entropy was used as a loss function. The model was trained by RMSprop optimizer for 100 epochs with the learning rate 0.01.
- **GA**: the GA Reader proposed in (Dhingra et al., 2017). It uses a novel multiplicative

Dataset	Method	Accuracy			
		Subtask 1	Subtask 2	Subtask 3-1	Subtask 3-2
Trial	MLP	0.44	0.28	0.20	0.19
	GA	0.32	0.25	0.24	0.25
	BERT-C	0.26	0.14	0.21	0.21
	BERT-S	<b>0.50</b>	<b>0.30</b>	<b>0.29</b>	<b>0.27</b>
Test	MLP	0.47	0.46	0.27	0.25
	GA	0.19	0.21	0.20	0.19
	BERT-C	0.38	0.34	0.24	0.25
	BERT-S	<b>0.50</b>	<b>0.49</b>	<b>0.34</b>	<b>0.39</b>

Table 3: Evaluation results on the trial and test sets of all subtasks

gating mechanism, combined with a multi-hop architecture to learn token embeddings based on the given question. The GA Reader obtains state-of-the-art results on three QA benchmarks. However, those benchmarks only focuses on concrete concept answering.

- **BERT-C**: a modified version of the proposed approach. Instead of using similarities between the fine-tuned placeholder and option embeddings, scalar outputs from an MLP are used. After the fine-tuned placeholder and option embeddings are obtained, each fine-tuned option embedding is concatenated to the fine-tuned placeholder embedding. Then, the concatenated embedding is fed to a MLP module. This MLP module consists of three dense layers with the output size 512, 128 and 1 respectively. The scalar outputs from all options are then concatenated and proceeded as in **BERT-S**. The other settings were also set as same as the settings in **BERT-S**.

## 5 Results

The proposed approach was evaluated on the trial and test sets of all three subtasks. The evaluation metric is accuracy. The results of subtask 1 and 2 are shown in Table 3. There are two results of subtask 3 shown as subtask 3-1 and subtask 3-2. Subtask 3-1 is the proposed approach trained on subtask 1 but evaluated on subtask 2. Similarly, subtask 3-2 is the proposed approach trained on subtask 2 but evaluated on subtask 1. As shown in the table, the proposed approach performed better in subtask 1 compared to subtask 2. It means that the pre-trained BERT model applied in this work is capable of understanding imperceptible words. In contrast, it is not suitable for nonspecific

abstract words as it performed poorer in subtask 2. For subtask 3, the proposed approach is not generalized across imperceptible and nonspecific concepts. This can be implied by the significant drop in accuracy in both subtask 3-1 and 3-2. However, compared with the baselines, the proposed approach performed better in all subtasks.

## 6 Conclusion

In this work, we propose to use the pre-trained BERT token embeddings for QA of abstract meaning. These embeddings are additional information that help understanding abstract meanings in the tasks. According to the results, our approach outperformed the baselines in every subtask. It means that the pre-trained BERT model is effective in QA of abstract meaning that focus on imperceptibility and nonspecificity. For the future work, it is worth to fine-tune the embeddings from the pre-trained model in an end-to-end manner. Other resources, e.g., semantic graphs, are also worth considering to provide more information for machine comprehension of abstract meanings.

## References

- Bin Bi, Chen Wu, Ming Yan, Wei Wang, Jiangnan Xia, and Chenliang Li. 2019. [Incorporating external knowledge into machine reading for generative question answering](#).
- Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. [A thorough examination of the CNN/Daily Mail reading comprehension task](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2358–2367, Berlin, Germany. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of](#)

deep bidirectional transformers for language understanding.

Bhuwan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. [Gated-attention readers for text comprehension](#).

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching machines to read and comprehend](#).

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. [RACE: Large-scale ReAding Comprehension dataset from Examinations](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized bert pretraining approach](#).

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#).

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter](#).

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).

Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.



# Noobs at Semeval-2021 Task 4: Masked Language Modeling for abstract answer prediction

Shikhar Shukla , Sarthak  
Samsung Research Institute  
Bangalore-560037, India

shikhar.00778@gmail.com, sarthak.j2709@gmail.com

Karm Veer Arya

ABV-Indian Institute of Information Technology & Management  
Gwalior-474015, India

kvarya@iiitm.ac.in

## Abstract

This paper presents the system developed by our team for Semeval 2021 Task 4: Reading Comprehension of Abstract Meaning. The aim of the task was to benchmark the NLP techniques in understanding the abstract concepts present in a passage, and then predict the missing word in a human written summary of the passage. We trained a Roberta-Large model trained with a masked language modeling objective. In cases where this model failed to predict one of the available options, another Roberta-Large model trained as a binary classifier was used to predict correct and incorrect options. We used passage summary generated by Pegasus model and question as inputs. Our best solution was an ensemble of these 2 systems. We achieved an accuracy of 86.22% on subtask 1 and 87.10% on subtask 2.

## 1 Introduction

There has been a lot of research in evaluating the performance of machine learning models to identify concrete concepts present in text and answer questions based on it (Hermann et al., 2015a). The organizers of ReCAM task at Semeval 2021 have provided a dataset to benchmark the models' performance on understanding the abstract concepts in the text in English language. The models are required to predict the missing words in a human written summary of the passage. This can help assess if the models can accurately capture the important concepts and meaning in the text.

The paper is organized as follows: In Section 2, we give a background on the problem and method that has been used, in Section 3, we present the proposed system architecture, in Section 4, we present the hyperparameters analysis done on the system, in Section 5, we present the results of the proposed architecture along with other approaches taken, and in Section 6, we conclude the paper.

## 2 Background

### 2.1 Dataset

The organizers provide two different datasets (Zheng et al., 2021) for two subtasks exploring two different definitions of abstractness (Spreen and Schulz, 1966; Changizi, 2008), *imperceptibility* and *nonspecificity*. Anything which can't be perceived is described as an *Imperceptible* concept (Example: culture, economy etc.) (Spreen and Schulz, 1966; Coltheart, 1981; Turney et al., 2011). *Nonspecificity*, as described by (Changizi, 2008), rather than looking at concrete things, focuses on generalizing the text (Example: hypernyms of words; vertebrate for whale). Subtask 3 explores the relationship between the two definitions.

### 2.2 Masked Language Modelling

Masked Language Models have played an important role in BERT (Devlin et al., 2019) and subsequent transformer models' success on different datasets. Masked Language Modelling is based on *Cloze* task (Taylor, 1953), which is described as filling the blanks in sentence using the surrounding context. Consider a Sequence  $\mathbf{S}$  containing  $n$  tokens ( $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \dots, \mathbf{w}_n$ ). In Masked Language Models, a token  $\mathbf{w}_t$  is replaced with a special token [MASK] and all the other tokens ( $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{t-1}, \mathbf{w}_{t+1}, \dots, \mathbf{w}_n$ ) are used to predict this token.

In our model, we use Roberta-Large's (Liu et al., 2019) Language Model (LMs), RobertaForMaskedLM, which randomly replaces 15% of tokens in a sequence and then tries to predict the masked word. Masked LMs perform better than left-to-right, right-to-left LMs, and concatenation of both. In a multi-layer Bidirectional LM, each word can indirectly see itself (from 2nd layer onwards) after the first layer, making the process redundant.

Recently, Pegasus model, pretrained on C4(Raffel et al., 2020) and HugeNews dataset, (Zhang et al., 2020) has shown state-of-the-art results for abstractive summarization on many summarization tasks (Narayan et al., 2018; Hermann et al., 2015b; Koupae and Wang, 2018). It has a transformer-based encoder-decoder architecture but has been trained with a novel self-supervised objective. It masks entire sentences in a corpus which are then generated later as one sequence capturing the abstractive summary.

### 3 System Overview

The key components in our proposed system are abstractive summarization of context using Pegasus, Roberta for Masked Language Modelling and Roberta for sequence classification. Algorithm 1 presents our system’s algorithm for predicting the option using the given context and masked question.

---

#### Algorithm 1 MLM + Sequence Classification

---

```

1: function PREDICTOPTION(context, question, options)
2:   summary_model ← Pegasus-XSum
3:   MaskedLM ← RobertaForMaskedLM(‘Large’)
4:   classifier ← RobertaForSequenceClassification(‘Large’)
5:   question ← question.replace(‘@placeholder’, ‘[MASK]’)
6:   context_summary ← summary_model(context)
7:   mlm_input ← concat(context_summary, question)
8:   top5_predictions ← MaskedLM(mlm_input)
9:   for i in top5_predictions do
10:    for j in options do
11:      if i = j then return i
12:    end if
13:  end for
14:  end for
15:  max_softmax ← 0
16:  answer ← 0
17:  for i in options do
18:    question ← question.replace(‘[MASK]’, i)
19:    input ← concat(context_summary, question)
20:    softmax_score ← classifier(input)
21:    if softmax_score[0] < softmax_score[1] then
22:      if softmax_score[1] > max_softmax then
23:        max_softmax ← softmax_score[1]
24:        answer ← i
25:      end if
26:    end if
27:  end for
28:  return answer
29: end function

```

---

**Abstractive Summarization** We use pretrained Pegasus-XSum (Zhang et al., 2020; Narayan et al.,

2018) to capture the abstractive summary from context which relates closely to the task at hand. We also experimented by finetuning it for the given task, by giving the corpus as input and passing the text, obtained from question after replacing @placeholder with the correct option, as output. We further experimented with extracting three line summaries by splitting context into three parts and extracting summary for each.

**RobertaForMaskedLM** The task at hand can be converted into predicting the masked token by replacing the @placeholder in question with the [MASK] token. We use RobertaForMaskedLM to predict the masked token. To help the model get the context in question, we prepend the summary from Pegasus-XSum to the masked question text. To finetune the model, we train it by passing concatenation of summary and masked question as input and providing concatenation of summary and filling the question with correct blank as output. Once the model is trained, we use Huggingface’s (Wolf et al., 2020) pipeline to predict the top 5 words for filling the masked token. Out of five predictions, word which has the highest probability of filling the blank and which is present in the given options is selected as the answer.

**Roberta For Sequence Classification** In some of the cases, we observed that the predicted words weren’t present in the options. For handling such cases, we use Roberta for Sequence Classification. We convert the task into a binary classification task by filling the masked question with correct option and treating it as one class and by filling the mask with wrong options as another class. This leads to a class-imbalanced dataset (1:4 ratio). To handle this, we randomly selected equal number of wrong-option class. On the test set, option which achieves the highest softmax score out of all the given options, is selected as the answer.

### 4 Experimental Setup

All the experiments were conducted in a Google Colab system with 12GB RAM and T4-Nvidia GPU. We experimented with different settings in three components in our system: summary extraction, Masked Language Model (MLM) and handling cases when word predicted by MLM is not present in the options.

[CLS] Super ##league leaders Manchester Thunder maintained their 100 % start to the season with victory over Surrey Storm . Hertfordshire Ma ##ver ##icks suffered only their second Super ##league [MASK] of the season after Team Bath beat them 55 - 54 in a thrill ##ing round 13 match . [SEP]

Visualizing the top 10 most important words.

Figure 1: Saliency map

**Summary Extraction** To identify the role of summary in the task, we experimented with three settings: no summary for prediction, extracting a one line summary from Pegasus-XSum and extracting a three line summary from Pegasus-XSum. To get a three line summary, we broke the context into three equal parts and fed into the summary model. Extracting one-line summary performed best for both the subtasks.

**Finetuning RobertaForMaskedLM** For Masked Language Model (MLM) to work, its very important for the model to identify and understand the context. Using the entire context meant that for some cases, the input size would exceed the limit (512 tokens) for model. So, we experimented with two settings, one line summary prepended to question, and using context when number of tokens wouldn't exceed the limit and one line summary when it did. For an untrained MLM, the latter setting performed better. Also, by training the MLM, a further improvement of 4-5% was observed. The MLM is trained with a batch size of 2 using the Adam (Kingma and Ba, 2017) optimizer with a learning rate of 1e-5.

**Handling missing cases for Masked Language Modelling** In few of the cases it was observed that the top 5 predictions made by MLM were not found in options. For such cases, we experimented with two settings: to find a pair of prediction and option which has the highest cosine similarity using Spacy embeddings (Honnibal et al., 2020), or to predict options using Sequence Classification (correct option as class 0, incorrect options as class 1). Input for Sequence classification was concatenating one line summary with question in which @placeholder is replaced with the options. If the option used to replace @placeholder is correct, output class is 0 else 1. Using Sequence Classification worked better than cosine similarity. Also, Roberta-Large outperformed other transformer models for

sequence classification on this dataset. The sequence classification model is trained with a batch size of 8, sequence length of 128 and using the Adam (Kingma and Ba, 2017) optimizer with a learning rate of 1e-5.

## 5 Results

We have mentioned the accuracy of all the systems developed by us for subtasks 1 and 2 in Table 1 and 2. We have also plotted a saliency map in Fig. 1 (with AllenNLP(Gardner et al., 2017) demo tool) to visualize the importance of each token in the prediction of the masked word. The example input from training set is: "Superleague leaders Manchester Thunder maintained their 100% start to the season with victory over Surrey Storm. Hertfordshire Mavericks [MASK] only their second Superleague loss of the season after Team Bath beat them 55 - 54 in a thrilling round 13 match." The top 5 predictions by the transformer model for the masked token are: "suffered", "recorded", "experienced", "sustained", "had". The correct option is "suffered". This demonstrates the effectiveness of using summary as context and formulating the problem as masked language modeling task.

### 5.1 Error Analysis

Across the 2 subtasks, masked language model made predictions which were present as part of options for 87% of the data. We present error analysis of the MLM over here:

#### a. Capturing more than one meaning

**Context: (truncated)** ... said he was too young to swim and should have still been in his mother's care ..... mother failed to show ... It tends to be when there's quite stormy weather the pups will get into trouble and they do get very tired, very hungry and very dehydrated and they just wouldn't survive without assistance. ....

**Question:** A @placeholder baby grey seal who was rescued from the rocks at Corbiere in Jersey will be flown to the UK on Friday .

System Description	Subtask 1	Subtask 2
Roberta-Large as binary classifier with summary and question as input	72.69	73.20
T5 large with passage, question, options concatenated	57.20	57.85
Untrained Roberta Large with MLM objective	72.64	73.36
Untrained Roberta Large with MLM objective and cosine similarity of word embeddings in failed cases	77.76	77.95
Trained Roberta Large with MLM objective	81.30	82.23
Trained Roberta Large with MLM objective and cosine similarity of word embeddings for failed cases	85.10	86.03
<b>Trained Roberta Large with MLM objective and Roberta binary classifier for failed cases</b>	<b>87.25</b>	<b>88.40</b>

Table 1: +-Accuracy of experimental setups on validation set

System Description	Trained On	Evaluated On	
		Subtask-1	Subtask-2
Trained Roberta Large with MLM objective and cosine similarity of word embeddings for failed cases	Subtask-1 Subtask-2	83.20 73.92	77.24 85.37
<b>Trained Roberta Large with MLM objective and Roberta binary classifier for failed cases</b>	Subtask-1 Subtask-2	<b>86.22</b> 78.56	82.44 <b>87.10</b>

Table 2: Accuracy of experimental setups on Test set

**Correct Option:** lone

**Predicted Options:** rare, stranded, distressed, tiny

For this particular example, the number of tokens didn't exceed 512 and entire context was used. As suggested by the context, its a rare event that a pup would be found without mother, stranded since mother didn't show or couldn't find the pup and is distressed as well.

#### b. Failure to capture information in one line summary

**Context: (truncated)** ... "I am very saddened by this, but what matters most now is the well-being of our kids," he told People magazine. "I kindly ask the press to give them the space they deserve during this challenging time." Jolie, 41, filed for divorce from Pitt, 52, citing irreconcilable differences on Monday. Her lawyer, Robert Offer, said the decision had been made "for the health of the family". .....

**Summary:** Actor Brad Pitt has said he is "very saddened" after his wife Angelina Jolie filed for divorce.

**Question:** Actor Brad Pitt has said he is " very saddened " that his wife Angelina Jolie has filed for divorce and has asked for @placeholder on their children 's behalf.

**Correct Option:** privacy

**Predicted Options:** support, custody, forgiveness, protection, counseling

For this particular example, 1-line summary was used and as evident no information pertaining to children could be located in it. Our best guess for these predictions are based on the data on which model is trained and is pretty much commonsense seeing the presence of "custody" and "support".

#### c. More than one correct answer

**Context:** . A lunar eclipse is when the Moon is fully covered by the Earth's shadow. It is the second one this year. The Moon's surface showed up coppery orange or red because the light from all the Earth's sunsets and sunrises were reflected on to it during the eclipse. In this timelapse, the Moon can be seen re-appearing as the shadow moves away.

**Question:** A total lunar eclipse has been visible across much of the Americas and Asia , resulting in a @placeholder " Blood Moon " .

**All Options:** bizarre, special, dramatic, lunar, visible

**Correct Option:** dramatic

**Predicted Options:** rare, special, spectacular, unique, partial

With a context being concise and straightforward, it can also be termed as special or rare other than dramatic. As per our algorithm, our system predicted special.

## 6 Conclusion

We have described the systems developed by us to solve the Reading Comprehension challenge at SemEval 2021. In our best performing submission, we framed the problem as a masked language modeling task. We used the predictions from a separately trained binary classifier when the above system failed to generate words which were not part of the options. Our models were able to achieve high accuracy with a relatively simple setup. We were ranked 11th out of 23 participants in subtask 1 and 12th out of 28 participants in subtask 2. As part of future work, we aim to use information from knowledge bases such as ConceptNet. This can help extract broader concepts related to the words predicted by the masked language model.

## References

- Mark A. Changizi. 2008. Economically organized hierarchies in WordNet and the Oxford English Dictionary. *Cognitive Systems Research*, 9(3):214–228.
- Max Coltheart. 1981. The MRC Psycholinguistic Database. *The Quarterly Journal of Experimental Psychology Section A*, 33(4):497–505.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
- Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015a. Teaching Machines to Read and Comprehend. *CoRR*, abs/1506.03340.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015b. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.
- Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization.
- Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A Large Scale Text Summarization Dataset.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach.
- Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don’t Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer.
- Otfried Spreen and Rudolph W. Schulz. 1966. Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.
- Wilson L. Taylor. 1953. “Cloze Procedure”: A New Tool for Measuring Readability. *Journalism Quarterly*, 30(4):415–433.
- Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and Metaphorical Sense Identification through Concrete and Abstract Context. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 680–690, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

# ZJUKLAB at SemEval-2021 Task 4: Negative Augmentation with Language Model for Reading Comprehension of Abstract Meaning

Xin Xie<sup>1,2\*</sup>, Xiangnan Chen<sup>1,2\*</sup>, Xiang Chen<sup>1,2\*</sup>, Yong Wang<sup>3</sup>,  
Ningyu Zhang<sup>1,2†</sup>, Shumin Deng<sup>1,2</sup>, Huajun Chen<sup>1,2†</sup>

<sup>1</sup> Zhejiang University & AZFT Joint Lab for Knowledge Engine

<sup>2</sup> Hangzhou Innovation Center, Zhejiang University <sup>3</sup> Microsoft

{xx2020, xnchen2020, xiang\_chen}@zju.edu.cn

{zhangningyu, 231sm, huajunsir}@zju.edu.cn

wangyon@microsoft.com

## Abstract

This paper presents our systems for the three Subtasks of SemEval Task4: Reading Comprehension of Abstract Meaning (ReCAM). We explain the algorithms used to learn our models and the process of tuning the algorithms and selecting the best model. Inspired by the similarity of the ReCAM task and the language pre-training, we propose a simple yet effective technology, namely, negative augmentation with language model. Evaluation results demonstrate the effectiveness of our proposed approach. Our models achieve the **4th** rank on both official test sets of Subtask 1 and Subtask 2 with an accuracy of 87.9% and an accuracy of 92.8%, respectively<sup>1</sup>. We further conduct comprehensive model analysis and observe interesting error cases, which may promote future researches.

## 1 Introduction

Past decades have witnessed the huge progress of representation learning in Natural Language Processing (NLP). With pre-trained language models, machine reading comprehension (MRC) models can extract answers from given documents and even yield better performance than humans on benchmark datasets such as Squad (Rajpurkar et al., 2016). However, these successes sometimes lead to the hype in which these models are being described as “understanding” language or capturing “meaning” (Bender and Koller, 2020). Note that the intention of MRC is letting the systems read a text like human beings, extracting text information and understanding the meaning of a text then answering questions, which means the systems can not only conclude the semantic of the text but also comprehend the abstract concepts under the constraint of

general knowledge regarding the world (Wang and Jiang, 2016). Nevertheless, little works as well as benchmarks focus on this direction.

**SemEval 2021 Task4** (Zheng et al., 2021) is an MRC task that focuses on evaluating the model’s ability to understand abstract words. Reading Comprehension of Abstract Meaning (ReCAM) task is divided into three Subtasks including **Subtask 1**: ReCAM-Imperceptibility, **Subtask 2**: ReCAM-Nonspecificity and **Subtask 3**: ReCAM-Intersection. Unlike previous MRC datasets such as CNN/Daily Mail (Hermann et al., 2015), SQuAD (Rajpurkar et al., 2018), and CoQA (Reddy et al., 2019) that request computers to predict concrete concepts, e.g. named entities. This task challenges the model’s ability to fill the abstract words removed from human-written summaries based on the English context.

Note that this task’s input format is similar to the MLM pre-training task of BERT (Devlin et al., 2019), which aims to predict the mask tokens. Pre-trained language models (PLMs) such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), DeBERTa (He et al., 2021) have achieved success on MRC tasks. Inspired by this, we introduce a simple yet effective method, namely, Negative Augmentation with Language model (NAL) in **SemEval 2021 Task4**. Specifically, we augment the answer distribution with an additional negative candidate from the mask language model’s prediction. Previous work (Petroni et al., 2019; Zhou et al., 2020) indicates that the pre-trained language model has already captured much world knowledge. Thus, we argue that knowledge can help guide the model training and identify those ambiguous abstract meanings. Further, we introduce other technologies such as label smoothing, domain-adaptive pre-training in our system. We describe the detailed approaches used for the Subtasks in Section 3.

Equal contribution and shared co-first authorship.

<sup>1</sup>Our implementation is publicly available at <https://github.com/zjunlp/SemEval2021Task4>

We conduct comprehensive experiments in Section 3, and we achieve the 4th system for Subtask 1: ReCAM-Imperceptibility and the 4th system for Subtask 2: ReCAM-Nonspecificity in the leaderboard. In our experiments, we observe that PLMs without fine-tuning can easily get 60+% accuracy on both Subtask 1 and Subtask 2, demonstrating that pre-trained language models already capture some abstract meanings. We further find that our negative augmentation with language model can improve the performance with 2.6% in Subtask 1 and 4.6% in Subtask 2. Finally, we conduct error analysis to promote future researches.

## 2 Background

Machine reading comprehension (MRC) has received increasing attention recently, which is a challenging task. According to the type of the answer, reading comprehension tasks can be divided into four categories (Chen, 2018): 1) Cloze-style: The question contains a "@placeholder," and the system must choose a word or entity from the set of candidate answers to fill in the "@placeholder" to make the sentence complete. 2) Multiple choice: In this type of task, Choosing a suitable answer from K sets of given answers. This answer can be one word or a sentence. 3) Span prediction: This kind of task is also called (Extractive question answering), which requires the system to extract a suitable range of text fragments from a given original text based on the question as to the answer. 4) Free-form answer: This task allows the answer to be any type of text, which is necessary to mine deep-level contextual semantic information according to a given question and a collection of candidate documents, and even combine multiple articles to give the best answer.

In **SemEval 2021 Task4**, it requires the system to have a strong ability of reading comprehension not only because the task is the cloze-style format as mentioned above but also the abstract words in answers. There are two definitions of abstract words: imperceptibility and nonspecificity. Concrete words refer to things, events, and properties that we can perceive directly with our senses (Spreeen and Schulz, 1966; Turney et al., 2011). Compared to concrete words like "trees" and "red," abstract words for imperceptibility are created by humans instead of pointing the things in the natural world. For example, as shown in Table 1, "want" and "achieve" means a person's attitude towards

---

<b>P:</b>	Briton Davies won F42 shot put gold with a Games record at Rio 2016, but was unable to defend his 2012 discus title as it did not feature in Brazil. "I don't normally say what I'm going for," said the Welshman, 25. "But this time <i>I'm definitely going for the two golds in both disciplines</i> and nothing will be better than being in front of a home crowd."
<b>Q:</b>	Paralympic champion Aled Sion Davies @placeholder two gold medals at the 2017 World Para Athletics Championships in London.
<b>A:</b>	(A) suffered (B) promoted (C) remains (D) <b>wants</b> (E) achieved

---

<b>P:</b>	... Low vitamin D levels can lead to brittle bones and rickets in children. The figures from the HSCNI show a dramatic rise in Vitamin D prescriptions over the last 10 years: The data does not include Vitamin D bought over the counter...
<b>Q:</b>	Rickets does not have the ring of a 21st Century problem - it sounds more like the @placeholder of a bygone era .
<b>A:</b>	(A) <b>horror</b> (B) size (C) fate (D) tale (E) death

---

Table 1: Examples of the **SemEval 2021 Task 4**. Given a passage and a question, the model needs to pick the best one from the five candidates to replace @placeholder.

something and a person's accomplishment about something. Meanwhile, the abstract words for non-specificity can be described as upper words. By determining whether one word can generalize another word, we can get dictionaries of different levels. The words with higher levels are the non-specificity words. Compared to concrete concepts like groundhog and whale, hypernyms such as vertebrate are regarded as more abstract (Changizi, 2008).

The difference between Subtask 1 and Subtask 2 is the definition of abstract words. So the input of both Subtask 1 and Subtask 2 are the same. The input of these tasks are shown in Table 1, it can be represented as a triple  $\langle P, Q, A \rangle$ , where  $P = s_1, s_2, \dots, s_m$  is the passage from CNN daily (Hermann et al., 2015),  $Q$  is a human-written summary based on the passage with one abstract word replaced by "@placeholder" and  $A$  is a set of candi-

date abstract words for filling in the ”@placeholder” in the question.

### 3 System Overview

#### 3.1 Model Design

Recently, with the development of the large Pre-trained Language Models (PLMs), such as GPT (Radford et al., 2018), BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), DeBERTa (He et al., 2021), have overwhelm the NLP community (Zhang et al., 2020c). The powerful semantic feature extraction capabilities of the PLMs make us only need to make better use of the BERT-like model itself for downstream tasks instead of adding different layers to the model.

Similar to the normal multi-choice task, we have five candidates, one passage, and one question per sample. Here we leverage PLMs as encoders to capture the global context representation about the passage, question, and answer. Then a decoder is used to determine the score of each  $\langle P, Q, A \rangle$  pair. Since we get  $A_1, \dots, A_n$   $n$  answers, for every passage, we construct  $n$  input samples as  $[Q - A_i; P]$ , the concatenation of  $Q - A_i$  and  $P$ . Because the question is the summary with an abstract word removed. We construct  $Q - A$  by replacing ”@placeholder” with the option from the candidate set instead of concatenating  $Q$  and  $A$ . After encoding all  $n$  inputs for a single passage, we get the global representations  $T_i$  for different options in the candidate set. During fine-tuning PLMs, the first special token [CLS] represents the global meaning of the whole input. We use an dense decoder layer to compute the score for all  $T_i$ , the calculation of score is as follow:

$$T_i = PLM(Q - A; P) \quad (1)$$

$$score_i = \frac{\exp(f(T_i))}{\sum_{i'} \exp(f(T_{i'}))} \quad (2)$$

where the  $[Q - A; P]$  is the input constructed according to the instruction of PLMs and MRC tasks, and the  $T_*$  is the final hidden state of the first token [CLS]. The candidate answers with higher scores will be identified as the final prediction.

Since previous research (Gao et al., 2020; Yang et al., 2019) demonstrate that there exists a gap between language model pre-training and fine-tuning the models in the downstream task and inspire by the similar task definition as MLM, we introduce

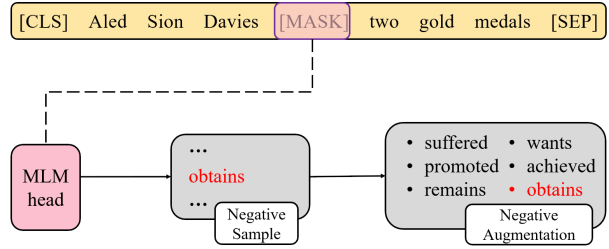


Figure 1: The procedure of Negative Augmentation with Language Model (NAL).

the negative augmentation with language model mechanism (Section 3.2). Note that the additional label will enhance the discriminability of the abstract meanings in a contrastive manner. In other words, the model is encouraged **NOT** to generate those abstract tokens from the language model, but the golden candidates from the given documents. We further introduce the label smoothing (Section 3.3), which can enhance the model performance. Finally, we leverage task-adaptive pre-training (Section 3.4) inspired by (Gururangan et al., 2020) to obtain better performance.

#### 3.2 Negative Augmentation with Language Model

Inspired by the same format of MLM and this task, we first conduct a toy experiment to test whether a PLM can get the right answer without any supervised signal. Firstly we replace the ”@placeholder” with [MASK] to reconstruct the input and ask the BERT model with MLM head to predict the word token at the [MASK]. Then we calculate the similarity between the word model predict and the options from the set of candidate answers. We set the option with the highest similarity score as the model’s choice. Then we find that the BERT model without any fine-tuning gets **60+%** accuracy in both Subtask 1 and Subtask 2. The result above shows that PLMs have the ability to predict abstract words, and those predicted words can be leveraged as negative candidates in the fine-tuning period.

Note that huge languages have quantities of parameters; the PLMs are able to store much knowledge through pre-training tasks. However, [MASK] is not used when fine-tuning the model for downstream tasks; how to use the knowledge stored by the model on pre-training tasks more explicitly on downstream tasks has become a hot topic of current research. Motivated by this, we try to bridge the gap between pre-train and downstream tasks. Inspired by the contrastive learning (Chen



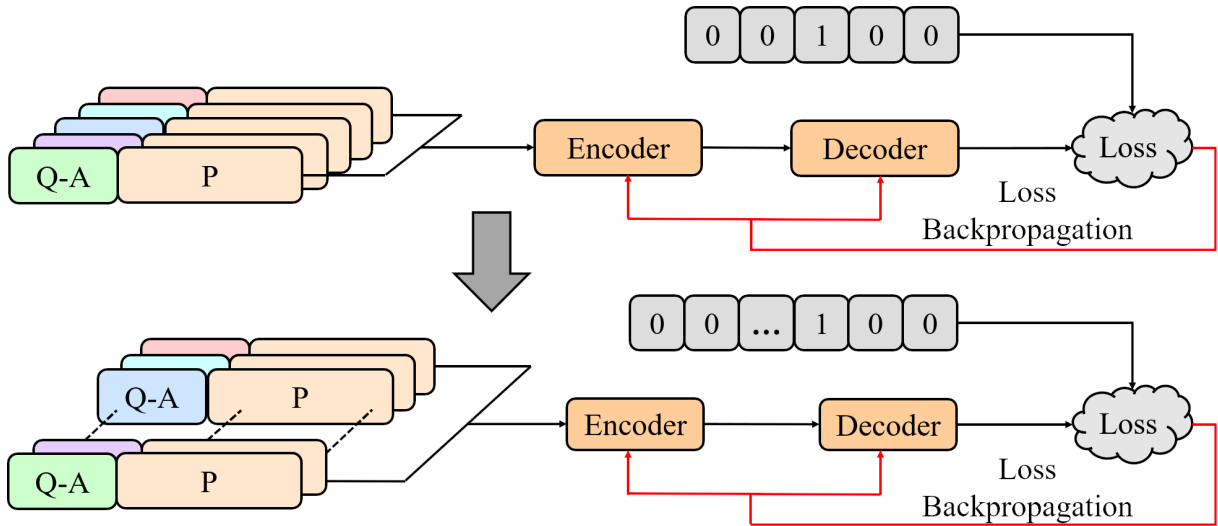


Figure 2: System overview (Best viewed in color.). The top of the Figure refers to the normal fine-tuning of multi-choice models, ignoring the form of pre-training tasks. While the bottom of the Figure refers to our system with Negative Augmentation with Language Model (NAL), which uses the abstract words predict by the original PLM as negative candidates to augment fine-tuning.

et al., 2020; Robinson et al., 2020) as stronger negative samples will help the model learning with better performance, we introduce our negative augmentation with language model method. Specifically, we let the PLMs predict the ”@placeholder” replaced with [MASK] token to generate negative candidates. Thus, we can leverage those negative words that may mislead the models to help train the models. Formally, we have:

$$P = p(m_i|\theta, [Q-A; P]), m_i \in [1, 2, \dots, |V|] \quad (3)$$

where  $P$  are the distribution of words model, predict,  $m_i$  is the token in the vocabulary, and  $|V|$  is the total number of the vocabulary. We can use the distribution to get the top confusing words to augment our models, which is described in Figure 2. Due to the limitation of GPU, we add the most possible word to augment our models.

### 3.3 Label Smoothing

Label smoothing is a well-known ”trick” to improve the model’s performance effectively. It encourages the activations of the penultimate layer to be close to the template of the correct class and equally distant to the templates of the incorrect classes (Müller et al., 2019). With more options than the original dataset by the approach mentioned in Section 3.2, label smoothing will magnify our method’s effect while fine-tuning the models. Suppose the output of the final layer and softmax layer

as follows:

$$p_k = \frac{e^{x^T w_k}}{\sum_{l=1}^L e^{x^T w_l}} \quad (4)$$

where  $p_k$  is the likelihood the model assigns to the  $k$ -th class,  $w_k$  represents the weights and biases of the last layer.  $x$  is the vector containing the activations of the penultimate layer of a neural network concatenated with ”1” to account for the bias. let us see the equitation about the cross entropy loss.

$$L = - \sum_{c=1}^M y_k \log(p_k) \quad (5)$$

The cross-entropy formula without Label smoothing only focuses on whether the positive example is true and does not pay attention to the negative examples’ relationship. We make the soft  $y$  as follows:

$$y_i = \begin{cases} (1 - \varepsilon), & \text{right answer} \\ \frac{\varepsilon}{K-1}, & \text{wrong answer} \end{cases} \quad (6)$$

We set  $\varepsilon$  as 0.1 in our models.

### 3.4 Task-Adaptive Pre-training

The BERT-like model is pre-trained in the general domain corpus such as Wikipedia. Since passages mainly come from CNN daily, the data distribution may be quite different from pre-training data. Therefore, we utilize task-adaptive pre-train BERT with masked language model and next sentence

Statistics / Dataset	Subtask 1	Subtask 2
# Train	3,227	3,318
# Trail	1,000	1,000
# Dev	837	851
# Test	2,025	2,017
Avg. # Length Per Passage	262	418

Table 2: Statistics of the **SemEval 2021 Task 4** dataset.

prediction tasks on the domain-specific data. Task-adaptive pre-training not only makes the model better fit the distribution in the domain but also helps the model to predict good negative words to enhance the original dataset, which is described in Section 3.2. We take two different approaches for task-adaptive pre-training as follows:

- 1) In-domain pre-training, we use the source data: CNN Daily to task-adaptive pre-training our base models(Sun et al., 2020).
- 2) Within-task pre-training, practically we replace the "@placeholder" with the correct answer and put the same input format as the fine-tuning steps, which is  $[Q - A; P]$  (Gururangan et al., 2020).

## 4 Experimental Setup

### 4.1 Dataset

In Subtask 1, the training/trail/development/test contains 3, 227/1, 000/837/2, 025 instances. In Subtask 2, the training/trail/development/test contains 3, 318/1, 000/851/2, 017 instances. The overall statistics can be found in Table 2.

### 4.2 Pre-processing

For data pre-processing, we use the byte-level BPE encoding (Sennrich et al., 2016), and the official vocabulary contains more than fifty thousand byte-level tokens. All tokens are stored in MERGES.TXT, while VOCAB.JSON is a byte-to-index mapping. Generally speaking, the higher the frequency, the smaller the byte index. Since the average length of the passage about Subtask 1 and Subtask 2 is 262 and 418, we divide those long context paragraphs. We limit the max number of tokens in an input sample  $[Q - A; P]$  to 256 for our system. Statically, 60% of the paragraphs exceeds the 256 tokens (including the special tokens like  $[CLS]$ ,  $[SEP]$  and so on. For these input samples, we divide them into new input samples with at most 256 tokens. To be more specific, we divide the passage to different inputs with the same question and answer.

## 4.3 Hyper-parameter Setting

Our system is implemented with PyTorch (Paszke et al., 2019) and we use the PyTorch version of the pre-trained language models<sup>2</sup>. We employ RoBERTa, ALBERT, and DeBERTa large models as our PLM encoder. We use AdamW optimizer (Loshchilov and Hutter, 2018) to fine-tune the models. We set the batch size to 1, and the max length of input to 256 for RoBERTa, 128 for ALBERT.

Usually, the batch size has a significant influence on the BERT-like model; due to the limit of GPU memory, we use gradient accumulation in our training steps. We set the gradient accumulation step as 32, which means the formal number of batch sizes is 32 in training. We pick the best learning rate from the dev set, fine-tuning the RoBERTa, ALBERT, DeBERTa with the learning rate of  $9 \times 10^{-6}$ ,  $1 \times 10^{-5}$  and  $1 \times 10^{-5}$  respectively. We set the number of epoch to 8 for ALBERT and 12 for RoBERTa and DeBERTa. Furthermore, we save the best model on the validation set for testing during training. Because the formats of both Subtask 1 and Subtask 2 are the same, we set the same batch size and max length of the input sequence for training.

## 5 Results

### 5.1 Subtask 1 Results

On Subtask 1, the ReCAM-Imperceptibility task, the evaluation results are illustrated in Table 3. We set the three baseline models: RoBERTa<sub>Large</sub>, DeBERTa<sub>Large</sub>, and ALBERT<sub>xxLarge</sub>. RoBERTa<sub>Large</sub> + NAL, DeBERTa<sub>Large</sub> + NAL, and ALBERT<sub>Large</sub> + NAL denotes the language model with our proposed negative augmentation with language model. Ensemble refers to the ensemble model of the three models as mentioned above with all strategies. We find that ALBERT achieves better performance in Subtask 1 but fails to get good performance in Subtask 2, while DeBERTa and RoBERTa have better performance in Subtask 2. Comparing with the original RoBERTa, DeBERTa, and ALBERT models, each model is hugely improved with NAL by about 2.1% accuracy. We further observe that DeBERTa and RoBERTa, which have the same architecture, obtain better performance than ALBERT in the dev and test sets. We think the possible reason is that ALBERT uses layer weight sharing, which

<sup>2</sup><https://github.com/huggingface/transformers> (version 3.3.0)

Model	Dev	Test
<i>Baseline</i>		
RoBERTa <sub>Large</sub>	83.3	-
ALBERT <sub>xxLarge</sub>	85.1	-
DeBERTa <sub>Large</sub>	84.1	-
<i>Ours</i>		
RoBERTa <sub>Large</sub> + NAL	85.9	86.1
ALBERT <sub>xxLarge</sub> + NAL	86.2	85.6
DeBERTa <sub>Large</sub> + NAL	86.7	86.8
Ensemble	<b>88.5</b>	<b>87.9</b>

Table 3: Results (Accuracy) on Subtask 1.

Model	Dev	Test
<i>Baseline</i>		
RoBERTa <sub>Large</sub>	86.7	-
ALBERT <sub>xxLarge</sub>	84.3	-
DeBERTa <sub>Large</sub>	87.7	-
<i>Ours</i>		
RoBERTa <sub>Large</sub> + NAL	91.1	89.7
ALBERT <sub>xxLarge</sub> + NAL	89.3	88.6
DeBERTa <sub>Large</sub> + NAL	91.3	90.3
Ensemble	<b>93.7</b>	<b>92.8</b>

Table 4: Results (Accuracy) on Subtask 2.

reduces the model’s generalization ability in reading comprehension, especially the abstract words meaning. Finally, the ensemble of the best model of RoBERTa, DeBERTa, and ALBERT lead to a significant improvement (**4.3%** accuracy) compared with baselines, which is also our final submission to the leaderboard.

## 5.2 Subtask 2 Results

On Subtask 2, the ReCAM-Nonspecificity task, the experiment results are showed in Table 4. Similar to the models in Subtask 1, we choose RoBERTa, DeBERTa and ALBERT as our baseline models. All RoBERTa<sub>Large</sub> + NAL , ALBERT<sub>xxLarge</sub> + NAL and DeBERTa<sub>Large</sub> + NAL are the models with negative augmentation with language model. Ensemble refers to the ensemble model of RoBERTa, DeBERTa, and ALBERT with all strategies. We notice that our proposed mechanism brings significant improvement (averaging **4.3%** of the accuracy score) compared with baselines, demonstrating the effectiveness of our proposed strategies such as negative augmentation with a language model, label smoothing, and task-adaptive pre-training. We observe

that ensemble approach of three enhanced models (RoBERTa<sub>Large</sub> + NAL, ALBERT<sub>xxLarge</sub> + NAL and DeBERTa<sub>Large</sub> + NAL) obtain the best accuracy of **92.8%** at test set, which is also our final submit to the leaderboard.

## 5.3 Subtask3 Results

Subtask3 focuses on the model’s transferability. During the evaluation period, we use the data on Subtask 2 to evaluate the models trained on the Subtask 1 and vice versa. We obtain the 82% accuracy of the model trained on Subtask 1 and evaluated on Subtask 2 on the dev set.

During experiments for all tasks, we have tried to use different decoders like MLP and other network architecture. Eventually, we find that it does not help to improve the system’s performance. An explanation is that the pre-trained language models (PLMs) have already captured global contextual sentence meaning at the [CLS] token.

## 5.4 Further Analysis

### 5.4.1 Analysis of Negative Augmentation with Language Model

During our experiments, we conduct case studies to figure out how our method of NAL helps the model to boost performance. From Table 5, we notice that the original PLM considers using the "all", "half" as its choice instead of "parts". Although fine-tuned on the downstream task, the baseline model still choose "half". In our NAL method, we add some misleading negative words to help models correct the knowledge learned from the pre-training task.

### 5.4.2 Analysis of Passage Length

In usual MRC tasks, the length of the passage is a key factor for the models to solve the problems. We conduct experiments to analyze the performance regarding different lengths of passage. Contrary to the common assumption, from Figure 3 and Figure 4, we observe that the instances with long passage obtain better performance. We think that abstract mean understanding may need comprehensive context information from the long sentence, and we will conduct further analysis in future works.

### 5.4.3 Case Study

We select four kinds of different types of error cases to promote further researches. We classify the examples according to the main causes (pre-training, fine-tuning, and so on) of the error. We

Example	
<b>Question:</b>	The Aurora Borealis, better known as the Northern Lights, was spotted across @placeholder of England on Sunday.
<b>Answer set</b>	{(A) millions, <b>(B) parts</b> , (C) half, (D) isle, (E) remains}
<b>NAL set</b>	{all, half, <b>parts</b> }
<b>Baseline</b>	(C) half
<b>Model with NAL</b>	(B) parts
<b>Question:</b>	The BBC is providing live coverage of the Scottish National Party conference in Glasgow. This live @placeholder has finished .
<b>Answer set</b>	{(A) results, (B) recording, <b>(C) event</b> , (D) action, (E) center}
<b>NAL set</b>	{blog, recording , stream}
<b>Baseline</b>	(B) recording
<b>Model with NAL</b>	(C) event

Table 5: We can clearly see the negative options can help the model better understand the abstract meaning in the passage and question. Answers are **bold** in the Table.

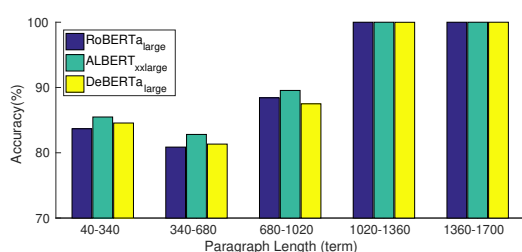


Figure 3: Results (Accuracy) on Subtask 1 with the length of passage.

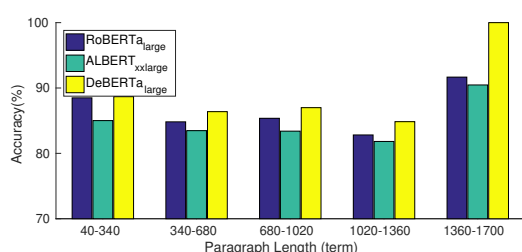


Figure 4: Results (Accuracy) on Subtask 2 with the length of passage.

think it will help us better understand what the model learns from pre-training and fine-tuning.

### Case 1 - Influenced by the original pre-training task

- **Passage:** "...found the United States to have the highest number of sleep deprived students, with 73% of 9 and 10 year olds and 80% of 13 and 14 year olds identified by their teachers as *being adversely affected*. The BBC's Jane O'Brien reports."
- **Question:** Sleep deprivation is a significant hidden factor in lowering the @placeholder

of school pupils , according to researchers carrying out international education tests .

- **Answer:** (A) morale (B) IQ (C) mortality (D) closure (E) achievement
- **Negative augmented choice:** (F) intelligence
- **Right Option:** (E) achievement
- **Wrong Option:** (B) IQ
- **Potential causes:** After pre-training on the large general domain corpus, PLMs have a huge bias on predicting the [MASK] token. Just like the "IQ" model predict in the "@placeholder". Even after fine-tuning, our models still cannot recognize the strong evidence "being adversely affected". In our daily life, we wouldn't hold that being adversely affected by lack of sleep can lead to a decrease in IQ. We usually say that lack of sleeping may lower one's achievement in the future.
- **How to help models?** To prevent the model from relying too much on pre-training tasks, we create more negative samples to help the model to understand what is wrong or right about the abstract words.

### Case 2 - Adverse affected by fine-tuning

- **Passage:** " 17 May 2017 Last updated at 12:44 BST Adrien Gulfo, wearing red, who plays for the Swiss side Pully Football, tried to clear the ball away from his goal with a spectacular bicycle kick. Unfortunately for him it all went very wrong - watch the video... There was a happy ending to the story for Gulfo though, Pully went through to the cup final on

penalties after the match finished 3-3.”

- **Question:** You won’t believe this own goal that was @placeholder in the Swiss lower league !
- **Answer:** (A) scored (B) born (C) eliminated (D) closed (E) beaten
- **Negative Augmented Choice:** (F) scored (model predict ”scored.” Because it is the right answer, so we choose another choice ”played” as an augmented choice. )
- **Right Option:** (A) scored
- **Wrong Option:** (E) beaten
- **Potential Causes:** It is quite weird that the original PLMs can predict the right answer, but fail to make it after fine-tuning. We suppose that in the process of fine-tuning, the inconsistency of abstract vocabulary prediction and the interference of other vocabulary caused the model’s effect in some cases to decrease instead.
- **How to help models?** We could use our approach of NAL to increase the weight of the knowledge learned in the pre-training task or leverage external knowledge (Zhang et al., 2019, 2020b; Yu et al., 2020; Zhang et al., 2020a).

### Case 3 - Obscure abstract word meaning

- **Passage:** ” ...Mr Habgood said: ”We’re pretty sure it will be popular because it was when East Street was closed for other reasons and we want to make it a friendlier place to be. ”It does fit with our larger objectives to improve the town and make it safer for cyclists and pedestrians.” ...”
- **Question:** Three busy town center streets are to be pedestrianised in a bid to improve @placeholder for shoppers and cyclists .
- **Answer:** (A) opportunities (B) services (C) quality (D) disruption (E) safety
- **Negative Augmented Choice:** (F) access
- **Right Option:** (E) safety
- **Wrong Option:** (B) services
- **Potential Causes:** Due the limit of GPU memory, we cannot put the long passage into the model once a time. So during the training, the model can only see a small chunk of the passage, so that it cannot get the global representation of the passage.
- **How to help models?** We chunk those long sentences with the approach of the sliding window to help the model understanding the

whole passage.

### Case 4 - Hypernyms is not always right

- **Passage:** ” North Wales Fire and Rescue Service was called to Express Linen Services on Vale Road in Llandudno Junction just before 19:30 GMT on Thursday. North Wales Police said a man was treated at the scene for smoke inhalation. Police have asked people to avoid the area...”
- **Question:** A number of @placeholder have been evacuated as firefighters tackle a blaze at a commercial laundry firm ’s premises in Conwy county.
- **Answer:** (A) families (B) properties (C) water (D) disruption (E) vehicles
- **Negative Augmented Choice:** (F) homes
- **Right Option:** (B) properties
- **Wrong Option:** (A) families
- **Potential Causes:** Hypernyms is the main focus of Subtask 2, the model may consider the ”families” as the upper level of the ”people” occur in the passage and choose the ”(A) families” instead of the right answer ”(B) properties”.
- **How to help models?** We try to use the proposed NAL to add more abstract words learned from the pre-training to mitigate this issue.

## 6 Conclusion

This paper presents our system design for the SemEval 2021 Task4. We propose a simple yet effective method called negative augmentation with language model. Comprehensive experiments demonstrate the effectiveness of our proposed approach. We also conduct case studies and investigate why the model fails to obtain the correct prediction.

Note that language models are pre-trained from the huge corpus; recently, researchers have identified the bias in the language model, which may mislead the model prediction. Our proposed negative augmentation with language model can help the model better discriminate candidates in fine-tuning, thus boost the performance. From another perspective, as depicts in Section 3.2, the language model without any fine-tuning gets **60+%** accuracy in both Subtask 1 and Subtask 2. This indicates that bias exists in the datasets (Part of the abstract meaning can be obtained from the language model). More strong benchmarks should be constructed in the future.

## 7 Acknowledgments

We want to express gratitude to the anonymous reviewers for their hard work and kind comments. This work is funded by 2018YFB1402800/NSFC91846204/NSFCU19B2027.

## References

- Emily M. Bender and Alexander Koller. 2020. [Climbing towards NLU: On meaning, form, and understanding in the age of data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, Online. Association for Computational Linguistics.
- Mark A. Changizi. 2008. [Economically organized hierarchies in WordNet and the Oxford English Dictionary](#). *Cognitive Systems Research*, 9(3):214–228.
- Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis, Stanford University.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. [A simple framework for contrastive learning of visual representations](#). In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. [Making pre-trained language models better few-shot learners](#). *CoRR*, abs/2012.15723.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. [Don’t stop pretraining: Adapt language models to domains and tasks](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [DeBERTa: Decoding-enhanced BERT with Disentangled Attention](#). *arXiv:2006.03654 [cs]*. ArXiv: 2006.03654.
- Karl Moritz Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. [Teaching machines to read and comprehend](#). In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A lite BERT for self-supervised learning of language representations](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Ilya Loshchilov and Frank Hutter. 2018. [Fixing Weight Decay Regularization in Adam](#).
- Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. 2019. [When does label smoothing help?](#) In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4696–4705.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. [Language models as knowledge bases?](#) In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. [Improving language understanding by generative pre-training](#).
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. [Know what you don’t know: Unanswerable questions for SQuAD](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. [Coqa: A conversational question answering challenge](#). *Trans. Assoc. Comput. Linguistics*, 7:249–266.
- Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. [Contrastive learning with hard negative samples](#). *CoRR*, abs/2010.04592.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural machine translation of rare words with subword units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Otfried Spreen and Rudolph W. Schulz. 1966. [Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns](#). *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.
- Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. [How to Fine-Tune BERT for Text Classification?](#) *arXiv:1905.05583 [cs]*. ArXiv: 1905.05583.
- Peter Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. [Literal and metaphorical sense identification through concrete and abstract context](#). In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 680–690, Edinburgh, Scotland, UK. Association for Computational Linguistics.
- Shuohang Wang and Jing Jiang. 2016. [Machine comprehension using match-lstm and answer pointer](#). *CoRR*, abs/1608.07905.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.
- Haiyang Yu, Ningyu Zhang, Shumin Deng, Hongbin Ye, Wei Zhang, and Huajun Chen. 2020. [Bridging text and knowledge with multi-prototype embedding for few-shot relational triple extraction](#). In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6399–6410. International Committee on Computational Linguistics.
- Ningyu Zhang, Shumin Deng, Juan Li, Xi Chen, Wei Zhang, and Huajun Chen. 2020a. [Summarizing chinese medical answer with graph convolution networks and question-focused dual attention](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, pages 15–24. Association for Computational Linguistics.
- Ningyu Zhang, Shumin Deng, Zhanlin Sun, Jiaoyan Chen, Wei Zhang, and Huajun Chen. 2020b. [Relation adversarial network for low resource knowledge graph completion](#). In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 1–12. ACM / IW3C2.
- Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. [Long-tail relation extraction via knowledge graph embeddings and graph convolution networks](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3016–3025, Minneapolis, Minnesota. Association for Computational Linguistics.
- Ningyu Zhang, Qianghuai Jia, Kangping Yin, Liang Dong, Feng Gao, and Nengwei Hua. 2020c. [Conceptualized representation learning for chinese biomedical text mining](#). *arXiv preprint arXiv:2008.10813*.
- Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. [SemEval-2021 task 4: Reading comprehension of abstract meaning](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.
- Xuhui Zhou, Yue Zhang, Leyang Cui, and Dandan Huang. 2020. [Evaluating commonsense in pre-trained language models](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9733–9740.

# PINGAN Omini-Sinitic at SemEval-2021 Task 4: Reading Comprehension of Abstract Meaning

Ye Wang    Yanmeng Wang    Haijun Zhu    Bo Zeng  
Zhenghong Hao    Shaojun Wang    Jing Xiao

Ping An Technology, Beijing 100191, China

{wangye430, wangyanmeng219, zhu haijun416, zengbo345,  
haozhenghong145, wangshaojun851, xiaojing661}@pingan.com.cn

## Abstract

This paper describes the winning system for subtask 2 and the second-placed system for subtask 1 in SemEval 2021 Task 4: Reading Comprehension of Abstract Meaning. We propose to use pre-trained ELECTRA discriminator to choose the best abstract word from five candidates. An upper attention and auto denoising mechanism is introduced to process the long sequences. The experiment results demonstrate that this contribution greatly facilitates the contextual language modeling in reading comprehension task. The ablation study is also conducted to show the validity of our proposed methods.

## 1 Introduction

Reading Comprehension of Abstract Meaning (ReCAM) (Zheng et al., 2021) is a cloze-style task, which takes a document and related human written abstract with one word replaced by a placeholder as input. The model is required to choose the best word from five candidates. The ReCAM consists of three subtasks. In subtask 1 and 2, participating systems are required to choose the best imperceptible concept and hyper-nyms concepts word respectively. Subtask 3 aims to evaluate performance of a system trained on one definition and test on the other.

Traditional cloze-style reading comprehension model (SA reader) (Kadlec et al., 2016) uses attention to directly pick the answer from the context, which makes model incapable to answer the question where the answer does not appear in passage. Furthermore, GA reader (Dhingra et al., 2017) adopts multi-hop attention mechanism to build query-specific representation of answer for ranking the candidates which is not part of passage.

Pre-trained language models (Radford et al., 2018; Devlin et al., 2019; Yang et al., 2019; Lan et al., 2020) have been widely adopted for context modeling in many Natural Language Processing

tasks. These models are pre-trained on huge corpora with plain texts and can better model contextual dependencies of tokens, thus enhance the performance of downstream approaches. As described in (Lai et al., 2017; Zhang et al., 2020; Chen et al., 2019; Zhu et al., 2018), they improved the performance of single-choice reading comprehension tasks by introducing pre-trained model, but they takes excessive memory for concatenating each option with the question and the passage.

GPT2 (Radford et al., 2018) has outperformed the SOTA result on cloze-style task CBT (Hill et al., 2016). As stated in (Radford et al., 2018), GPT2 computes the probability of each choice and the rest of the sentence conditioned on this choice according to the pre-trained model, the answer is the choice with highest probability. GPT2 outperforms RNN-based models without fine-tuning on CBT task, we assume that pre-trained language model has better potential to address the cloze-style problem than fine-tuning the pre-trained model with an additional ranking network.

The most relevant work to our model is Pattern-Exploiting Training (PET) (Schick and Schütze, 2020a,b), which proposed to reformulate the sentence classification task to cloze-style task with defined golden answer word as supervising signal. The comparison between PET and our proposed method is reported in section 5.

Different with PET, We propose a novel Auto Denoising Discriminator for Abstract Concept in reading comprehension (ADDAC) by fine-tuning the pre-trained discriminator of ELECTRA (Clark et al., 2020). Auto Denoising is introduced while processing long sequences. By fine-tuning the pre-trained model on its own structure with the original pre-training loss, the tasks results is significantly improved even with small train dataset, we suppose the representations stored in the pre-trained model has been maximum reserved in this way.



## 2 Background

### 2.1 Task Description

The task intends to answer a cloze-style question, the answer to which depends on the understanding of a context document provided with the question. The model is also provided with a set of possible answers from which the correct one is to be selected. This can be formalized as follows:

The training data consists of tuples  $(q, d, a, A)$ , where  $q$  is an abstract sentence of document  $d$  and one word in  $q$  is replaced with a placeholder.  $A$  is a set of possible options and  $a \in A$  is the golden answer. Both  $q$  and  $d$  are sequences of words and golden answer  $a$  does not appear in the article  $d$ .

### 2.2 ELECTRA vs ALBERT

Since the success of BERT (Devlin et al., 2019), pre-trained language models have adopted a large amount of parameters to achieve better modeling performance. ALBERT (Lan et al., 2020) uses factorized embedding parameterization and cross-layer parameter sharing to greatly reduce the amount of model parameters and achieved SOTA in multiple natural language understanding task. ALBERT outperforms other pre-trained language models which is trained by MLM(masked language model) in combination with PET method for this task.

ELECTRA (Clark et al., 2020) proposed the RTD (replaced token detection) task with adversarial learning as an alternative to the MLM task. A smaller generator is used to replace the special token [MASK] in training samples, and then a discriminator is trained to predict each word in the input is real or generated by generator. In section 3.1, we will elaborate the details about our proposed discriminator mechanisms based on ELECTRA. The performance comparison between ALBERT with PET and our proposed discriminator model on ReCAM task is reported in Section 5.

## 3 System overview

### 3.1 Pre-trained Discriminator

We approach the competition tasks as cloze-style task, which can be reformulated to masked language modeling (Devlin et al., 2019) problems. As shown in Figure 1, we replace placeholder with golden and negative options in question  $q$  denoted as  $q_A$ , which is further concatenated with corresponding document  $d$  in pre-trained model input

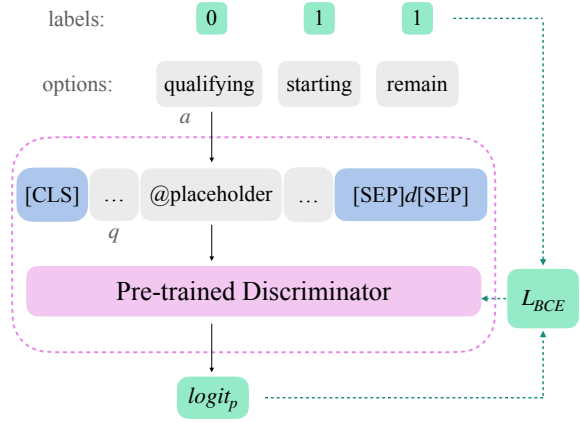


Figure 1: Discriminator overview, the placeholder in  $q$  is replaced by  $a$  which is one of candidate options, the label of golden option is 0 followed the ELECTRA pre-training setting.

style ( $[CLS]q_A[SEP]d[SEP]$ ). We ignore the part of sequence which exceed the maximum length of input sequence. The input sequence is forward to ELECTRA discriminator and the hidden states are calculated by Equation 1

$$H_{q_A} = F([q_A; d]) \quad (1)$$

$$\text{logit}_p = D(H^p) \quad (2)$$

where  $F$  is the pre-trained 24-layer transformers and  $D$  is a linear layer which classify the hidden states of replaced word from ELECTRA Discriminator.  $H_{q_A} \in \mathbb{R}^{N \times d}$  are the hidden states of input sequence, in which  $N$  and  $d$  are the maximum length of input sequence and the dimension of hidden states. Then we only use the hidden states of placeholder  $H^p \in \mathbb{R}^d$  selected from  $H_{q_A}$  as the input to  $D$ .  $BCE$  (Binary Cross Entropy) loss, which measures the Binary Cross Entropy between the golden label and the output, is used for binary classification as Equation 3.

$$L_{BCE} = BCE(\text{Sigmoid}(\text{logit}_p), l_p) \quad (3)$$

where  $l_p$  is the binary label of option word (replacing placeholder in input sequence). The label is set to 0 for golden option, 1 for negative options, which is the same as ELECTRA pre-training setting (Clark et al., 2020). While in inference, the option with lowest  $\text{logit}_p$  is regard as the right answer to the question. The experiment results show that discriminator outperforms the ranking and PET implementation on this task (see section 5).

We also implement a ranking model based on ELECTRA for single-choice reading comprehension task to compare with the Discriminator approach. The question and document is combined with each option as the input of ELECTRA encoder, which is denoted as  $S \in \mathbb{R}^{N \times m}$ , where  $N$  is the length of sequences and  $m$  is the number of options. A linear layer and a softmax layer is added after ELECTRA encoder as the ranking network, which use the hidden states of [CLS] in  $S$  as input.

### 3.2 Processing Long Input Sequence

Most of the pre-trained models have a limitation in processing long sequences. The maximum sequence length of ELECTRA is 512, which is much shorter than the maximum length of input sequence (i.e. concatenation of question and document). We propose two methods for processing long input sequences: 1) document is segmented to shorter passages, which leads to the problem of mislabel samples. We introduce an auto denoising mechanism to address the problem. 2) We adopt an upper attention upon transformers.

**Auto Denoising** The whole document  $d$  is segmented into a set of fragments  $\{s_1^d, s_2^d, \dots, s_k^d\}$  with a fixed window size. Then, these fragments combine  $q_a$  to form model input sequence. In the inference phase, the lowest predicted logit is selected from all results of fragments as Equation 4.

$$\text{logit}_p = \min_{i=1}^k D(F([q_A; s_i^d])) \quad (4)$$

where  $q$  with placeholder replaced by a specific option from  $A$  denote as  $q_A$ . However, this method causes the noisy-label problem. Supposed that the answer  $a$  just finds proof from fragment  $s_1^d$ , which results in other samples  $(0, [q_a; s_{i \neq 1}^d])$  to be mislabeled samples, which significantly impact training of model and decrease the prediction accuracy. Therefore, we take the advantage of a noise-tolerant loss (bi-tempered logistic loss, BT (Amid et al., 2019)) and a noise detection method (over-fitting to under-fitting, O2U (Huang et al., 2019)) in this work. The BT logistic loss lowers gradient on noisy samples which relieve the negative effects on model training via adjusting bi-temperature. The O2u makes full use of the property that model is easier to forget the mislabeled samples than the clean samples, to identify and filter the mislabeled samples.

**Upper Attention** The long sequence of input is segmented into small segments with the length of 512 tokens and each segments are concatenated with same question to form the input sequences, which are encoded into hidden states  $H_i \in \mathbb{R}^{d \times 512}$ , where  $i$  is the index of segments of passage, the  $d$  is the dimension of hidden states and 512 is the sequence length. The hidden states of placeholder is denoted as  $H_i^p \in \mathbb{R}^d$ . We use a 1-layer multi-head self-attention block to fuse the hidden states of placeholder from multiple segments output.

$$H_{fuse}^p = A_g(H_1^p, H_2^p \dots H_k^p) \quad (5)$$

where  $k$  is the number of segments,  $A_g$  is 1-layer multi-head-attention, without residual connection.  $H_{fuse}^p$  is applied in Equation 2 and Equation 3 for training.

### 3.3 Optimizer

The ELECTRA-large which has large amount of parameters tends to over-fit on small training dataset. We utilize RecAdam (Chen et al., 2020) to fine-tune the pre-trained model to address the over-fitting problem. RecAdam optimizer is proposed to address the catastrophic forgetting problem of sequential transfer learning paradigm by introducing a recall and learn mechanism into Adam optimizer, which maintain the learned knowledge in pre-trained model while learning a new task.

### 3.4 Data augmentation

To further boost the performance of our proposed model, we conduct data augmentation. We random pick 3000 articles in CNN/DailyMail (Hermann et al., 2015), and crawl 824 latest articles from BBC news website. The CNN news is much longer than the training samples, while the length of BBC news is approximately same. The extra training samples are generated in following steps: 1) The title of news article is used as abstract. 2) We pick one most meaningful word from abstract by TF-IDF scores and the origin word is used as golden option. 3) We use words with same category of POS (Part Of Speech) from other documents as negative options. We train models on the extra to dataset as warming up and further train the models with training dataset. Unfortunately, extra training data did not effectively improve the performance of our model on this task. We just use extra data in models ensemble phase.

## 4 Experimental setup

### 4.1 Data

We use the official released dataset of SemEval2021 Task4 for experiments. The dataset of subtask 1 contains 3227/837/2025 samples for train, dev and test data. The subtask 2 dataset contains 3318/851/2017 samples for train, dev and test data. The maximum/mean length of subtask 1 and subtask 2 training data are 2275/374.34 and 2274/578.31, respectively. The statistics of sample length shows that length of article in subtask 2 is much longer than the length in subtask 1. The rate of the sequences exceeding 512 tokens is 13.95% in subtask 1, 42.46% in subtask 2, which may cause that the methods for processing long sequences are more effective in subtask 2.

Since the provided dataset is small, we apply data augmentation in model ensemble to further improve generalization of the model (see Section 3.4).

### 4.2 Parameter settings

Our implementation is based on the Pytorch framework for transformer-based models(Wolf et al., 2020). We trained our model based on the pre-trained ELECTRA-Large discriminator, and adopt the same model structure for subtask 1 and subtask 2. We use Adam optimizer with a learning rate of 1e-5, batch-size of 32 to train the baseline model, which is actually the ELECTRA discriminator. The max sequence length is 512 and the epoch of training is set to 5. To address the overfitting problem, we apply RecAdam with sigmoid annealing function, where the annealing rate is 0.01 and the annealing time-step is 500. The coefficient of the quadratic penalty is set to be 5,0000. For Auto Denoising, the two temperatures and label smoothing of BT are equal to 0.9, 1.5 and 0.1 respectively. The maximum learning rate, minimum learning rate and epochs in cyclical round about O2U are set to be 5e-5, 1e-6 and 5.0 respectively.

Since only 5 submissions are permitted in submitting phase, we trained multiple models under different settings for model ensemble. We also adopt 8-fold cross-validation training to improve the model generalization.

### 4.3 Ensemble

Two strategies are used for our final submissions on test data: 1) we ensemble all 8 models from 8-fold cross-validation training by averaging their outputs,

which is trained on the train data of each subtask; 2) we trained multiple models on the train data and the augmented data with different model structures. 7 top different models are selected based on the dev accuracy for models ensemble, then average their outputs as the final output. Moreover, the model trained in cross-validation is Discriminator with Auto Denoising and RecAdam optimizer for subtask 2, and Discriminator with RecAdam for subtask 1. While in top ensemble, the techniques of Discriminator, Auto Denoising, RecAdam and Upper Attention are applied in different models.

## 5 Results and Analysis

### 5.1 Single Model Performance

We implement two baseline models for comparison with our proposed method. The ALBERT PET is the combination of PET method (Schick and Schütze, 2020a,b) and ALBERT-xxlarge model, which is trained by the MLM. The golden answer of training dataset is used as the target for MLM, but the negative ones are omitted in training. The ELECTRA Rank reformulates the cloze-style task to single-choice task, which is described in section 3.1.

Label accuracy is the official metric of the tasks and Table 1 shows the results on development dataset. Task3(1-2) is the subtask 3 which is trained on subtask 1 dataset and test on subtask 2, Task3(2-1) means train on subtask 2 and test on subtask 1. It is obvious that our proposed ELECTRA Discriminator significantly improve the performance of all tasks and outperform the best baseline by 4.7%/1.16%/3.76%/8.85% in subtask1/subtask2/subtask3(1-2)/subtask3(2-1) respectively. This confirms our hypothesis that pre-trained language model has more potential for cloze-style task. The PET with ALBERT is not suitable for this task because it can not utilize the negative options. The ELECTRA Rank performance is unsatisfactory, suggesting that fine-tuning on ranking network damage the knowledge stored in the pre-trained model.

The results of ablation study are also reported in Table 1. Disc (Discriminator) with RecAdam achieves further improvement in all tasks by 0.25% / 0.53% / 0.25% / 0.26% in subtask1 / subtask2 / subtask3(1-2) / subtask3(2-1) respectively, which prove the RecAdam optimizer is more effective for pre-trained model and also promotes the model generalization. Disc+Upper,

Table 1: Single Model Performance on dev dataset, the ablation study is demonstrated below.

Method	Task1	Task3(1-2)	Task2	Task3(2-1)
ALBERT PET	89.25	83.31	90.48	82.80
ELECTRA Rank	88.53	88.13	90.24	83.75
ELECTRA Discriminator	93.42	90.71	93.88	91.16
+RecAdam	<b>93.66</b>	91.19	94.12	91.40
+Upper	93.54	89.54	<b>94.35</b>	91.39
+Upper+RecAdam	93.31	<b>91.42</b>	94.12	91.51
+AutoDenoise+RecAdam	93.55	<b>91.42</b>	94.01	<b>91.88</b>

Table 2: Ensemble Performance on dev and test dataset, where 8-fold is the models from 8-fold cross-validation training, top ensemble means that ensemble the models with top dev accuracy.

	Method	Task1	Task3(1-2)	Task2	Task3(2-1)
Dev set	8-fold ensemble	92.47	-	92.94	-
	top ensemble	94.50	-	95.53	-
Test set	8-fold ensemble	<b>93.04</b>	93.90	94.99	91.35
	top ensemble	92.74	94.19	<b>95.29</b>	91.65

Disc+AutoDenoise+RecAdam achieve the best results in task2 and task3(2-1) respectively. This prove the validity of the two methods we proposed for long sequences. The Upper Attention achieve the best result on task2, while AutoDenoise achieve the best result on task3. We ensemble them to produce a stronger system. The Upper and AutoDenoise do not effectively improve the baseline in subtask 1, since the mean length of subtask 1 is 374.34 which is much shorter than the max sequence length of original pre-trained model.

## 5.2 Ensemble Performance

The performances of ensemble models are shown on Table 2, which is obtained from the competition leader-board. Our system got the first place in subtask 2 and the second place in subtask 1. For the subtask 3, our ranking is first place in subtask3(2-1), and second place in subtask3(1-2), that indicates our system has strong transfer capability in abstractive reading comprehension tasks. Ensemble results on dev dataset are exhibited for comparison, and we do not experiment model ensemble on dev data of subtask 3.

## 5.3 Memory-Efficiency

We trained all compared models on 16GB Tesla-V100 GPU, except for ELECTRA Rank which takes 17GB with batch size set to 1. Our proposed Discriminator only take 9GB, since one option with article and question is considered as single training

sample for binary classification task. In contrast, ELECTRA Rank is required to encode the input sequence with different options at once to learn the ranking function between positive and negative options. ELECTRA Discriminator is much faster than ELECTRA Rank to converge. The epoch of training ELECTRA Discriminator is less than 5 and ELECTRA Rank needs at least 10 train epochs to converge.

## Conclusion

In this paper, we propose an effective framework of combining ELECTRA discriminator with denoising learning method to boost the performance of cloze-style reading comprehension task. Our proposed model outperforms all others participating system on subtask 2 and gets the second-placed on subtask 1. We have conducted an ablation study, demonstrating the validity of Discriminator, Upper attention and Auto denoising. Pre-trained models have made great performance gain compared to traditional neural network models in many natural language tasks and is able to build comprehensive hidden representation of input text. The above experiment results may suggest that the current pre-trained model mechanism still has room for improvement.

## References

- Ehsan Amid, Manfred K. Warmuth, Rohan Anil, and Tomer Koren. 2019. [Robust bi-tempered logistic loss based on bregman divergences](#). In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14987–14996.
- Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. [Recall and learn: Fine-tuning deep pretrained language models with less forgetting](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7870–7881. Association for Computational Linguistics.
- Zhipeng Chen, Yiming Cui, Wentao Ma, Shijin Wang, and Guoping Hu. 2019. Convolutional spatial attention model for reading comprehension with multiple-choice questions. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6276–6283.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.
- Bhuvan Dhingra, Hanxiao Liu, Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. 2017. Gated-attention readers for text comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1832–1846.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701.
- Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children’s books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Jinchi Huang, Lie Qu, Rongfei Jia, and Binqiang Zhao. 2019. [O2u-net: A simple noisy label detection approach for deep neural networks](#). In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3325–3333. IEEE.
- Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 785–794.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. [Language models are unsupervised multitask learners](#).
- Timo Schick and Hinrich Schutze. 2020a. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676.
- Timo Schick and Hinrich Schutze. 2020b. It’s not just size that matters: Small language models are also few-shot learners. *CoRR*, abs/2009.07118.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*

2019, *NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. 2020. DCMN+: dual co-matching network for multi-choice reading comprehension. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9563–9570.

Boyuan Zheng, Xiaoyu Yang, Yuping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. SemEval-2021 task 4: Reading comprehension of abstract meaning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*.

Haichao Zhu, Furu Wei, Bing Qin, and Ting Liu. 2018. Hierarchical attention flow for multiple-choice reading comprehension. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6077–6085.

# NEUer at SemEval-2021 Task 4: Complete Summary Representation by Filling Answers into Question for Matching Reading Comprehension

Zhixiang Chen<sup>1†</sup>, Yikun Lei<sup>1†</sup>, Pai Liu<sup>1,2†</sup>, Guibing Guo<sup>1\*</sup>

<sup>1</sup>Northeastern University, Shenyang, China

<sup>2</sup>Westlake University, Hangzhou, China

<sup>†</sup> These three authors contributed equally.

{zxchen8830, ethanlei9931, pailiu1998}@gmail.com  
guogb@swc.neu.edu.cn

## Abstract

SemEval task 4 aims to find a proper option from multiple candidates to resolve the task of machine reading comprehension. Most existing approaches propose to concat question and option together to form a context-aware model. However, we argue that straightforward concatenation can only provide a coarse-grained context for the MRC task, ignoring the specific positions of the option relative to the question. In this paper, we propose a novel MRC model by filling options into the question to produce a fine-grained context (defined as summary) which can better reveal the relationship between option and question. We conduct a series of experiments on the given dataset, and the results show that our approach outperforms other counterparts to a large extent.

## 1 Introduction

In order to make the computer understand, represent and express better, we study the ability of MRC to understand Abstract definitions (Spren and Schulz, 1966; Changizi, 2008) in the Reading Comprehension of Abstract Meaning task (ReCAM). In ReCAM task, there are two kinds of abstract definitions, one is imperceptibility and the other is nonspecificity. To evaluate the model performance comprehensively, three subtasks are designed (Zheng et al., 2021):

Subtask 1) ReCAM-Imperceptibility - A passage, a question with a placeholder, and answers used to fill in the placeholder are given. The answers are all imperceptibility words so as to evaluate the model's ability to comprehend these imperceptibility words.

Subtask 2) ReCAM-Nonspecificity - A passage, a question with a placeholder, and answers used to fill in the placeholder are given. The answers are all Nonspecificity words so as to evaluate the

model's ability to comprehend these Nonspecificity words.

Subtask 3) To provide more insights into the relationship between two abstract views, it requires to test the performance of the model trained on one data set and evaluated on the other one.

Because question is obtained by hollowing out summary of passage and answer is abstract words to fill in (Zheng et al., 2021), there is a semantically complementary relationship between the question and answer. As shown in the Figure 1, in the previous work (Jin et al., 2020; Zhu et al., 2020), the model inputs passage, question and answer independently. It can only provide a coarse-grained context for the MRC task, ignoring the specific positions of the option relative to the question. Question cannot be a complete sentence, and the answer lacks contextual information about the question. To solve this problem, we introduce **Complete Summary Representation by Filling Answers into Question for Matching Reading Comprehension (ComSR)**. ComSR convert the original question and answer into summary. In this way, we can turn the problem into matching correct summary for the passage. By replacing the answer option into the question placeholder, the original question has higher semantic integrity, and the original answer obtains the context information from the question to eliminate language ambiguity such as polysemy.

Based on the results, ComSR is superior to pre-trained language models and the latest MRQA models, with the accuracy rate of 64.76% in subtask 1, 64.86% in subtask 2, and shows great generalization ability in Subtask 3.

## 2 System overview

Figure 2 illustrates the overall structure of ComSR. Since just one summary option belongs to the pas-

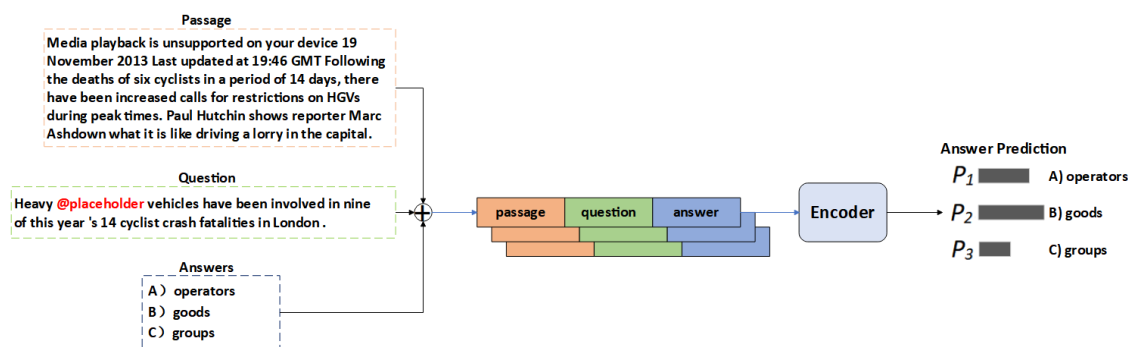


Figure 1: The framework of previous MCQA model. The abstract word in answer has a polysemy phenomenon. Entering question and answer separately cannot eliminate ambiguity.

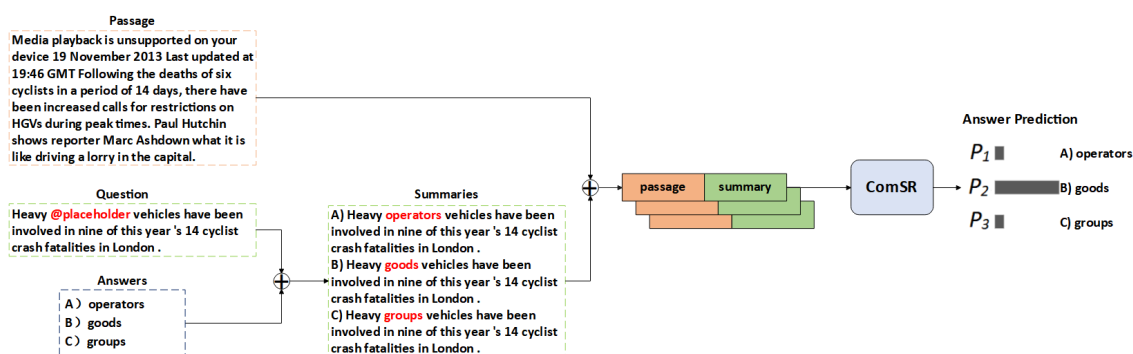


Figure 2: The framework of our model. By replacing the answer option into the question placeholder, answer obtains the context information from question, so the ambiguity is eliminated.

sage, we input passage and summary together to better reflect the corresponding relationship between the passage and the summaries. As BERT has been shown to be a powerful feature extractor for various tasks (Ran et al., 2019), we use BERT as the encoder to extract the vector representation of the semantic meaning of the passage and the summaries, as well as the relationship between the passage and the summaries. Finally, we use a classifier layer to convert the hidden state into the final selection of the model.

## 2.1 Input Layer

The initial question uses ”@placeholder” to represent the abstract word, which makes the question sentence not a complete sentence. We replace the ”@placeholder” in question with the answer word to get the summary  $S$ . As a complete sentence,  $S$  provides rich context information to answer, thus effectively eliminating the ambiguity caused by polysemy in answer. Then the passage  $P$  and the summary  $S$  are concatenated, using [CLS] at the beginning and [SEP] at the end of sentences to get input vector.

$$x = [p; s] \quad (1)$$

where  $x$  denotes input vector,  $p$  denotes passage vector and  $s$  denotes summary vector.

## 2.2 Encoder

Although Pre-trained language models have achieved good results in various NLP tasks, most existing MCQA datasets are small in size, pre-trained language models are hard to get adequate training. To solve this problem, we use the encoder obtained through multi-stage multi-task learning in MMM (Jin et al., 2020) as the initial encoder<sup>1</sup>. The reasons are as follows. First, the use of multiple data sets for training makes up for the shortcomings of the small amount of data in a single data set. It uses four similar task data sets, MultiNLI (Williams et al., 2017), SNLI (Young et al., 2014), DREAM (Sun et al., 2019), and RACE (Lai et al., 2017) for training, which has significantly more data than a single MCQA data set, thus solving the defect of insufficient data in a single data set. Second, the model uses two natural language inference datasets, MNLI and SNLI, as out-of-domain

<sup>1</sup>[https://drive.google.com/drive/folders/1EECS9na9PpX9CO\\_cCzYj9FDkiBvOpyxv](https://drive.google.com/drive/folders/1EECS9na9PpX9CO_cCzYj9FDkiBvOpyxv)



source datasets for Coarse-tuning, and two MCQA datasets, DREAM and RACE, as in-domain source datasets for Multi-task learning. The characteristics of different data sets make the model have better generalization ability (Jin et al., 2020). Third, because these data sets are highly related to ReCAM, it is suitable to use MMM encoder as initial encoder of ComSR.

$$\mathbf{H}^p = Bert(\mathbf{p}), \quad \mathbf{H}^s = Bert(\mathbf{s}) \quad (2)$$

where  $Bert(\cdot)$  indicate BERT model which return the last attention layer.  $\mathbf{H}^p \in R^{l \times |p|}$ ,  $\mathbf{H}^s \in R^{l \times |s|}$  are sequence representation of the passage and summary.  $|p|$ ,  $|s|$  are the length of the passage and summary.  $l$  is the dimension of the hidden state.

### 2.3 Answer Prediction

In the answer prediction module, we pass the output of the encoder through a fully connected layer and a pooling layer to finally get the score of each option.

$$s_k = \mathbf{v}_s^T \tanh(\mathbf{W}_k [\mathbf{H}^p; \mathbf{H}^s] + \mathbf{b}_k) \quad (3)$$

where  $s_k$  is score of the answer,  $\mathbf{v}_s^T$  and  $\mathbf{W}_k$  are learnable parameters.  $\tanh(\cdot)$  performs tanh activation function,  $\mathbf{W}_k \in \mathbf{R}^{l \times l}$ ,  $\mathbf{b}_k \in \mathbf{R}^l$ ,  $\mathbf{v}_s^T \in \mathbf{R}^l$ .

The probability  $P(k | P, S)$  of summary  $S_k$  to be the correct answer is computed as

$$P(k | P, S) = \frac{\exp(s_k)}{\sum_{i=1}^{N=5} \exp(s_i)} \quad (4)$$

Then according to  $P(k | P, S)$ , the loss function is defined as

$$J(\theta) = -\log(P(k | P, S)) \quad (5)$$

## 3 Experimental setup

### 3.1 Dataset

The data set of Reading Comprehension of Abstract Meaning (ReCAM) is provided by the organizer of the competition containing a large number of abstract words in answers. Abstract words refer to thoughts and concepts that are far from immediate perception. In ReCAM, we divide abstract words into Imperceptibility words and Nonspecificity words, and provide two data sets, respectively. The accuracy we showed in the paper was tested on the dev dataset.

#### 3.1.1 ReCAM-Imperceptibility

Imperceptible words are the words of being Imperceptibility by eyes or senses, such as experience, success, significant, challenge. This data set provides a passage, a question with a placeholder and five answers with an imperceptibility word. The training set/test set/validation set contains 3227/2025/837 pieces of data respectively.

#### 3.1.2 ReCAM-Nonspecificity

Nonspecific words are words with very broad concepts, and hypernyms are often words of this type, such as food, jewelry, people, and vehicle. This data set provides a passage, a question with a placeholder and five answers with a Nonspecificity word. The training set/test set/validation set contains 3318/2017/851 pieces of data respectively.

### 3.2 Implementation Details

In the test method, we use the accuracy of the model in the answer option of the data set as the measurement standard. For model training, we will train each tested model for 10 epochs at a learning rate of  $3e-05$ , use cross-entropy to calculate loss, and use Adam optimizer (Kingman and Ba, 2015) for fine-tuning. On hardware devices, we use GeForce RTX-2080Ti to provide computing.

## 4 Results

### 4.1 Comparison with Baselines

Since our model uses BERT as encoder, we use the original BERT (Devlin et al., 2018) for comparison. As pre-trained language models like BERT, we use ALBERT (Lan et al., 2019) and RoBERTa (Liu et al., 2019) in the experiment. In addition, we also adopt MMM (Jin et al., 2020) that performed well in multi-choice data sets such as RACE (Lai et al., 2017) and DREAM (Sun et al., 2019) to do experiments.

Model	Subtask	
	1(%)	2(%)
BERT-base	47.19	50.65
RoBERTa	22.34	22.44
ALBERT	36.91	37.25
MMM+BERT-base	57.11	59.11
ComSR+BERT-base(ours)	<b>64.76</b>	<b>64.86</b>

Table 1: Experimental results on ReCAM. The overall best results are in bold face.

Model	I→N (%)	N→I (%)
ComSR(Passage + summary)(ours)	50.65(14.11 ↓)	51.73(13.13 ↓)
ComSR(Passage + summary + question)	48.53(17.18 ↓)	49.94(15.51 ↓)
ComSR(Passage + summary + answer)	44.89(18.55 ↓)	46.71(18.38 ↓)
MMM(Passage + question + answer)	39.48(17.63 ↓)	43.73(15.38 ↓)

Table 2: Comparison generalization on ReCAM. I→N denotes training on the Imperceptibility dataset and testing on the Nonspecificity dataset, N→I is the opposite. ↓ represents the drop for a model compared to the test in its own test set and + denotes concatenation.

Table 1 show that the model has similar features in task1 and task2. In the experimental results of task1 and task2, we can find that (1) Although pre-trained language models have slightly different results depending on the models, the overall level is low on the small data set. (2) MMM proposed for the MCQA task performs better than the pre-trained language model on ReCAM. This is because the encoder of MMM uses the Multi-stage Multi-task Learning to train on NLI data sets such as MNLI and SNLI and on MCQA data sets such as DREAM and RACE, which is more suitable for ReCAM. (3) The performance of ComSR is better than other baselines, and it has significantly improved. It can be seen that, under the condition of using the same encoder, replacing the placeholder of question with answer is more conducive to understand sentence meaning.

## 4.2 Analysis Studies

### 4.2.1 The completeness of the semantic expression of summary

Converting question and answer into summary input improves semantic integrity and the model’s ability to understand sentences. However, is the semantics provided by summary complete? To study this problem, we set up the following experiment.

Model	Subtask 1(%)	Subtask 2(%)
ComSR(Passage + summary)(ours)	64.76	64.86
ComSR(Passage + summary + question)	65.71	65.45
ComSR(Passage + summary + answer)	63.44	65.09

Table 3: Comparison among different implementation of the input method on ReCAM. + denotes concatenation.

If there is a loss of semantics in the synthesis of the summary, then the summary can be supplemented when the original question and answer are used, thereby improving the performance of the model. However, it can be seen from the table 3 results that some of the experimental results of the passage+ summary+answer and passage+summary+question groups are equal to passage+summary and even slightly lower than passage+summary. Therefore, the conversion of question and answer into summary makes the semantics of options more complete, and there is no semantic loss.

### 4.2.2 Research on model generalization

To study the generalization ability of the model, (1) we use the ReCAM-Imperceptibility data set to train and test it in the ReCAM-Nonspecificity data set. (2) we use the ReCAM-Nonspecificity data set to train and test it in the ReCAM-Imperceptibility data set. In order to study the most suitable input form, we use passage+summary+question, passage+summary+answer and MMM (passage+question+answer) model for comparison.

According to the experimental results in table 2, after switching the data set for testing, the highest accuracy is ComSR(Passage + summary). Although the results of the test in another data set have dropped, the ComSR using the Passage and summary received the least impact. In summary, ComSR(Passage + summary) performs best in the above models, and the method of matching summary and Passage alone has stronger generalization ability.

### 4.2.3 Case Analysis

Table 4 shows a passage and its corresponding question and answers. To understand ComSR’s ability to disambiguate, we specifically selected samples that contained polysemous words in answers. In this example, the answer ”goods” is a typical polysemous word.

<b>Passage</b>	Media playback is unsupported on your device 19 November 2013 Last updated at 19:46 GMT Following the deaths of six cyclists in a period of 14 days, there have been increased calls for restrictions on HGVs during peak times. Paul Hutchin shows reporter Marc Ashdown what it is like driving a lorry in the capital.
<b>Question</b>	Heavy @placeholder vehicles have been involved in nine of this year 's 14 cyclist crash fatalities in London.
<b>Answers</b>	(a) operators (b) goods (c) groups (d) patrol (e) air

Table 4: Example of ReCAM.

Models take the logarithm of the value to get the predictions. Because many of the predictions are negative, we add bias = 12.5 to the predictions for easy viewing and show it in Figure 3.

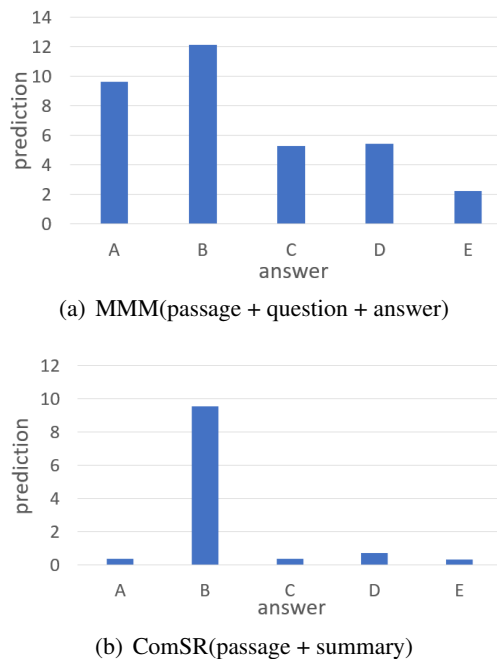


Figure 3: Prediction distributions in different models. The model selects the answer with the highest predicted value as the output option.

We observe that although MMM selects the right option, the difference between the correct answer and other option values in the probability distribution is not obvious. On the contrary, correct

prediction value is much higher than other values in ComSR. By combining question and answer into summary, ComSR can not only improve the semantic integrity of the sentence, but also eliminate the ambiguity caused by the polysemy of the original answer. Therefore, even for polysemous words that are prone to ambiguity, ComSR can obtain accurate answers by fully understanding the summary representation.

## 5 Conclusion

We proposed ComSR for multiple choice reading comprehension to complete summary representation. Merging the separated question and answer into a summary could improve semantic integrity and better reveal the relationship between option and question. Results showed that our model can achieve relatively high performance compared to the latest baseline in terms of accuracy and generalization ability.

## References

- Mark A Changizi. 2008. Economically organized hierarchies in wordnet and the oxford english dictionary. *Cognitive Systems Research*, 9(3):214–228.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Di Jin, Shuyang Gao, Jiun-Yu Kao, Tagyoung Chung, and Dilek Hakkani-tur. 2020. Mmm: Multi-stage multi-task learning for multi-choice reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8010–8017.
- DP Kingman and J Ba. 2015. Adam: A method for stochastic optimization. conference paper. In *3rd International Conference for Learning Representations*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

- Qiu Ran, Peng Li, Weiwei Hu, and Jie Zhou. 2019. Option comparison network for multiple-choice reading comprehension. *arXiv preprint arXiv:1903.03033*.
- Otfried Spreen and Rudolph W Schulz. 1966. Parameters of abstraction, meaningfulness, and pronounciability for 329 nouns. *Journal of Verbal Learning and Verbal Behavior*, 5(5):459–468.
- Kai Sun, Dian Yu, Jianshu Chen, Dong Yu, Yejin Choi, and Claire Cardie. 2019. Dream: A challenge data set and models for dialogue-based reading comprehension. *Transactions of the Association for Computational Linguistics*, 7:217–231.
- Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.
- Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.
- Boyuan Zheng, Xiaoyu Yang, Yu-Ping Ruan, Quan Liu, Zhen-Hua Ling, Si Wei, and Xiaodan Zhu. 2021. Semeval-2021 task 4: Reading comprehension of abstract meaning.
- Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Dual multi-head co-attention for multi-choice reading comprehension. *arXiv preprint arXiv:2001.09415*.

# WLV-RIT at SemEval-2021 Task 5: A Neural Transformer Framework for Detecting Toxic Spans

Tharindu Ranasinghe<sup>1</sup>, Diptanu Sarkar<sup>2</sup>, Marcos Zampieri<sup>2</sup>, Alexander Ororbia<sup>2</sup>

<sup>1</sup>University of Wolverhampton, UK

<sup>2</sup>Rochester Institute of Technology, USA

T.D.RanasingheHettiarachchige@wlv.ac.uk

## Abstract

In recent years, the widespread use of social media has led to an increase in the generation of toxic and offensive content on online platforms. In response, social media platforms have worked on developing automatic detection methods and employing human moderators to cope with this deluge of offensive content. While various state-of-the-art statistical models have been applied to detect toxic posts, there are only a few studies that focus on detecting the words or expressions that make a post offensive. This motivates the organization of the SemEval-2021 Task 5: Toxic Spans Detection competition, which has provided participants with a dataset containing toxic spans annotation in English posts. In this paper, we present the WLV-RIT entry for the SemEval-2021 Task 5. Our best performing neural transformer model achieves an 0.68 F1-Score. Furthermore, we develop an open-source framework for multilingual detection of offensive spans, i.e., MUDES, based on neural transformers that detect toxic spans in texts.

## 1 Introduction

The widespread adoption and use of social media has led to a drastic increase in the generation of abusive and profane content on the web. To counter this deluge of negative content, social media companies and government institutions have turned to developing and applying computational models that can identify the various forms of offensive content online such as aggression (Kumar et al., 2018, 2020), cyber-bullying (Rosa et al., 2019), and hate speech (Ridenhour et al., 2020). Prior work has either designed methods for identifying conversations that are likely to go awry (Zhang

et al., 2018; Chang et al., 2020) or detecting offensive content and labelling posts at the instances level – this has been the focus in the recent shared tasks like HASOC at FIRE 2019 (Mandl et al., 2019a) and FIRE 2020 (Mandl et al., 2020), GermEval 2019 Task 2 (Struß et al., 2019), TRAC (Kumar et al., 2018, 2020), HatEval (Basile et al., 2019a), OffensEval at SemEval-2019 (Zampieri et al., 2019b) and SemEval-2020 (Zampieri et al., 2020).

With respect to identifying offensive language in conversations, comments, and posts, noticeable progress has been made with a variety of large, annotated datasets made available in recent years (Pitenis et al., 2020; Rosenthal et al., 2020). The identification of the particular text spans that make a post offensive, however, has been mostly neglected (Mathew et al., 2021) as current state-of-the-art offensive language identification models flag the entire post or comment but do not actually highlight the offensive parts. The pressing need for toxic span detection models to assist human content moderation, processing and flagging content in a more interpretable fashion, has motivated the organization of the SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021).

In this paper, we present the WLV-RIT submission to the SemEval-2021 Task 5. We explore several statistical learning models and report the performance of the best model, which is based on a neural transformer. Next, we generalise our approach to an open-source framework called MUDES: Multilingual Detection of Offensive Spans (Ranasinghe and Zampieri, 2021a). Alongside the framework, we also release the pre-trained models as well as a user-friendly web-based User Interface (UI) based on Docker, which provides the functionality of automatically identifying the offensive spans in a given input text.

WARNING: This paper contains text excerpts and words that are offensive in nature.

## 2 Related Work

**Datasets** Over the past several years, multiple post-level, offensive language benchmark datasets have been released. In [Zampieri et al. \(2019a\)](#), the authors compiled an offensive language identification dataset with a three-layer hierarchical annotation scheme – profanity, category, and target identification. [Rosenthal et al. \(2020\)](#) further extended the dataset using a semi-supervised model that was trained with over nine million annotated English tweets. Recently, [Mathew et al. \(2021\)](#) released the first benchmark dataset which covered the three primary areas of online hate-speech detection. The dataset contained a 3-class classification problem (hate-speech, offensive, or neither), a targeted community, as well as the spans that make the text hateful or offensive. Furthermore, offensive language datasets have been annotated in other languages such as Arabic ([Mubarak et al., 2017](#)), Danish ([Sigurbergsson and Derczynski, 2020](#)), Dutch ([Tulkens et al., 2016](#)), French ([Chiril et al., 2019](#)), Greek ([Pitenis et al., 2020](#)), Portuguese ([Fortuna et al., 2019](#)), Spanish ([Basile et al., 2019b](#)), and Turkish ([Çöltekin, 2020](#)).

Apart from the dataset released for SemEval-2021 Task 5, HateXplain ([Mathew et al., 2021](#)) is, to the best of our knowledge, the only dataset that we could find that has been annotated at the word level. The dataset consists of 20,000 posts from Gab and Twitter. Each data sample is annotated with one of the hate/offensive/normal labels, communities being targeted, and words of the text are marked by the annotators who support the label.

**Models** In the past, trolling, aggression, and cyberbullying identification tasks on social media data have been approached using machine and deep learning-focused models ([Kumar et al., 2018](#)). Across several studies ([Malmasi and Zampieri, 2017, 2018](#); [Waseem and Hovy, 2016](#)) researchers have noted that  $n$ -gram based features are very useful when building reliable, automated hate-speech detection models. Statistical learning models aided with natural language processing (NLP) techniques are frequently used for post-level offensive and hateful language detection ([Davidson et al., 2017](#); [Indurthi et al., 2019](#)). Given the increased use of deep learning in NLP tasks, offensive language identification has seen the introduction of methods based on convolutional neural networks (CNNs) and Long Short-term Memory

(LSTM) networks ([Badjatiya et al., 2017](#); [Gambäck and Sikdar, 2017](#); [Hettiarachchi and Ranasinghe, 2019](#)). The most common approach has been to use a word/character embedding model such as Word2vec ([Mikolov et al., 2013](#)), GloVe ([Pennington et al., 2014](#)), or fastText ([Mikolov et al., 2018](#)) to embed words/tokens and then feed them to an artificial neural network (ANN) ([Zampieri et al., 2019b](#)).

With the introduction of BERT ([Devlin et al., 2019](#)), neural transformer models have become popular in offensive language identification. In hate speech and offensive content identification in Indo-European languages, the BERT model has been shown to outperform GRU (Gated Recurrent Unit) and LSTM-based models ([Ranasinghe et al., 2019](#)). In [Mandl et al. \(2019b\)](#), the best performing teams on the task employed BERT-based pre-trained models that identified the type of hate and target of a (text) post.

The SemEval-2019 Task 6 ([Zampieri et al., 2019b](#)) presented the challenge of identifying and categorizing offensive posts on social media, which included three sub-tasks. In sub-task A: offensive language identification, [Liu et al. \(2019a\)](#) applied a pre-trained BERT model to achieve the highest F1 score. In Sub-task B: automatic categorization of offense types, BERT-based models also achieved competitive rankings. We noticed similar trends in SemEval-2020 Task 12 ([Zampieri et al., 2020](#)) as well. Not limited to English, transformer models have yielded strong results in resource-scarce languages like Bengali ([Ranasinghe and Zampieri, 2020](#)) and Malayalam ([Ranasinghe et al., 2020](#)) along with cross-lingual transfer learning from resource-rich languages ([Ranasinghe and Zampieri, 2020, 2021b](#)). Nonetheless, despite the recent success of statistical learning in offensive language detection problems, due to the lack of finer-grained, detailed datasets, models are limited in their ability to predict word-level labels.

## 3 Task and Dataset

In the SemEval-2021 Task 5 dataset, the sequence of words that makes a particular post or comment toxic is defined as a *toxic span*. The dataset for this task is extracted from posts in the Civil Comments Dataset that have been found to be toxic. The practice dataset has 690 instances out of which 43 instances do not contain any toxic spans. The training dataset has a total of 7,939 instances and

Post	Offensive Spans
Stupid hateries have completely fucked everything	[0, 1, 2, 3, 4, 5, 34, 35, 36, 37, 38, 39]
Victimitis: You are such an asshole.	[28, 29, 30, 31, 32, 33, 34]
So is his mother. They are silver spoon parasites.	[]
You're just silly.	[12, 13, 14, 15, 16]

Table 1: Four comments from the dataset along with their annotations. The offensive words are displayed in red and the spans are indicated by the character position in the instance.

comprises 485 instances without any toxic spans. Each instance is composed of a list of toxic spans and the post (in English). In Table 1, we present four randomly selected examples from the training dataset along with their annotations.

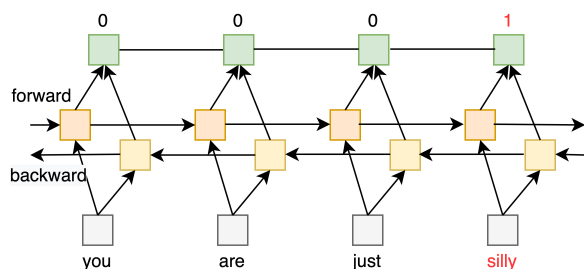


Figure 1: The Bi-LSTM-CRF model. Green squares represent the top CRF layer. Non-offensive and offensive tokens are shown as 0 and 1, respectively.

## 4 Methodology

### 4.1 Lexicon-based Word Match

Lexicon-based word-matching algorithms often achieve balanced results. For the lexicon, we collected profanity words from online resources<sup>1,2</sup>. Then, we added the toxic words present in the training dataset and we run a simple word matching algorithm the trie data structure. As anticipated, the algorithm does not evaluate the toxic spans contextually and misses censored swear words. For instance, the word  $f^{**}k$  is missed, which is not present in the lexicon. Nonetheless, this result provides as a useful baseline performance measurement for the task.

### 4.2 Recurrent Networks: Long Short-Term Memory

Long Short-term Memory (LSTM) is a recurrent neural network model that uses feedback connec-

<sup>1</sup><https://www.cs.cmu.edu/~biglou/resources/bad-words.txt>

<sup>2</sup><https://github.com/RobertJGabriel/Google-profanity-words>

tions to model temporal dependencies (past-to-present) in sequential data. Bidirectional LSTM (Bi-LSTM) is capable of learning contextual information both forwards and backwards in time compared to conventional LSTMs. In this study, we used the Bi-LSTM architecture given this bidirectional ability to model temporal dependencies. Conditional random fields (CRF) (Lafferty et al., 2001) are a statistical model that are capable of incorporating context information and are highly used for sequence labeling tasks. A CRF connected to the top of the Bi-LSTM model provides a powerful way to model relationships between consecutive outputs (across time) and provides a means to efficiently utilize past and future tag information to predict the current tag.

The final hybrid model is comparable to the previous state-of-the-art sequence tagging Bi-LSTM-CRF model (Huang et al., 2015). Figure 1 presents the Bi-LSTM-CRF architecture we designed for this study, which has 4.2 million trainable parameters. We trained the model on mini-batches of 16 samples with a 0.005 learning rate for 5 epochs with a maximum sequence length of 200.

### 4.3 Neural Transformers

Recently, pre-trained language models have been shown to be quite useful across a variety of NLP tasks, particularly those based on bidirectional neural transformers such as BERT (Devlin et al., 2019; Li et al., 2019). Transformer-based models have also been shown to be highly effective in sequence classification tasks such as named entity recognition (NER) (Luoma and Pyysalo, 2020). In our work, we extend the BERT model by integrating a token level classifier. The token-level classifier is a linear transformation that takes the last hidden state of the sequence as the input and produces a label for each token as its output. In this case, each token will be predicted to have one of two possible labels – toxic or not toxic. We fine-tuned the uncased BERT transformer model with a maximum

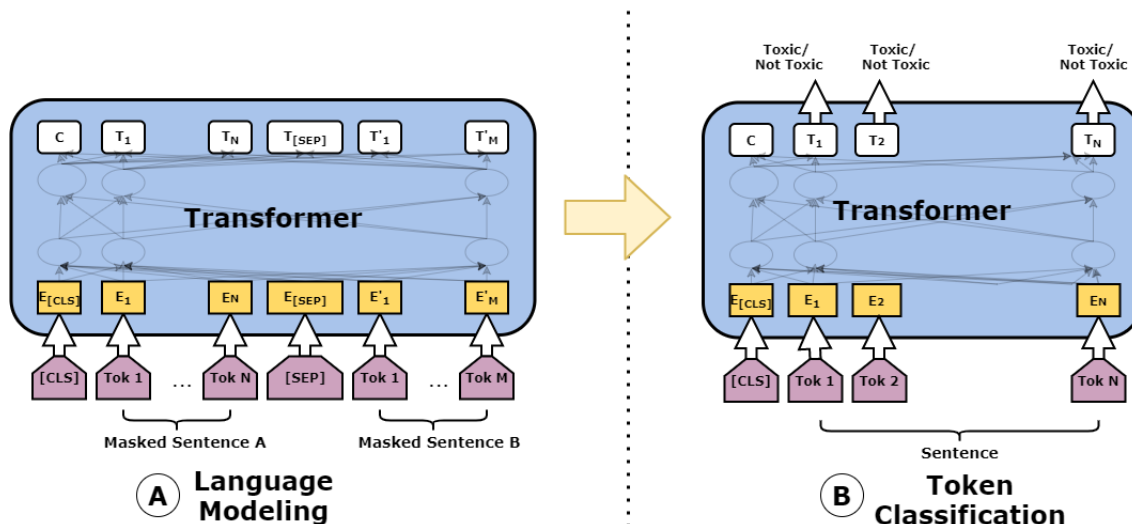


Figure 2: The two-part model architecture. Part A depicts the language model and Part B is the token classifier. (Ranasinghe and Zampieri, 2021a)

sequence length of 400 with batches of size of 16.

We also experimented with customising the layers in between the BERT transformer and token-classification layer by adding a CRF layer between them given that it has been shown that BERT-CRF architectures often outperform BERT baselines in similar sequence labeling tasks (Huang et al., 2019; Souza et al., 2020). Therefore, we added a sequential CRF layer on top of the BERT transformer and further incorporated dropout (probability of dropping a neuron was 0.2) to introduce some regularization. Unfortunately, in our experiments, we found that adding a CRF layer does not significantly improve the final generalization results. Additionally, we experimented with transfer learning to identify if a further boost in model generalization was possible if we first trained a basic BERT transformer on HateXplain (Mathew et al., 2021) and then fine-tuned it using our extended architecture as described above. However, the transfer learning process did not improve results any further.

**Development of MUDES** Given the success we observed using neural transformers such as BERT, we developed a (software) framework we call MUDES (Ranasinghe and Zampieri, 2021a): Multilingual Detection of Offensive Spans, an open-source framework based on transformers to detect toxic spans in texts. MUDES offers several capabilities in addition to the (automatic) token classification we described earlier. MUDES has the following components: *a) Language Modeler*: Fine-tuning transformer models using masked language

modeling before performing the downstream task often leads to better results (Ranasinghe and Hettiarachchi, 2020) and MUDES incorporates this, *b) Transformer Type Variety*: since there are many varieties of neural transformers, e.g., XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019b) that have been shown to outperform BERT-based architectures (Ranasinghe and Hettiarachchi, 2020; Hettiarachchi and Ranasinghe, 2020a), our software framework provides support for these architectures, and, finally, *c) Model Ensembling*: multiple MUDES models with different random seeds can be trained and the final model prediction is the majority vote from all the models, aligning with the approach taken in Hettiarachchi and Ranasinghe (2020b, 2021); Jauhainen et al. (2021).

The complete architecture of MUDES is depicted in Figure 2. We used several popular transformer models including BERT (Devlin et al., 2019), XLNET (Yang et al., 2019), RoBERTa (Liu et al., 2019b), SpanBERT (Joshi et al., 2020), and ALBERT (Lan et al., 2020). We compared these transformer architectures against the spaCy token classifier baseline (reported by the competition organisers) and report these results in Section 5. Since adding a CRF layer did not improve the results in our models, we do not add this to MUDES.

Parameter optimization involved mini-batches of 8 samples using the Adam update rule (global learning rate was  $2e-5$  and a linear warm-up schedule over 10% of the training data was used). Models were evaluated using a validation subset that contained 20% of the training data. Early stopping



was executed if the validation loss did not improve over 10 evaluation steps. Models were trained for 3 epochs on an Nvidia Tesla K80 GPU using only the training set provided.

## 5 Evaluation and Results

For evaluation, we followed the same procedure that the task organisers have used to evaluate the systems.

Let system  $A_i$  return a set  $S_{A_i}^t$  of character offsets for parts of a text post that have been found to be toxic. Let  $G_t$  be the character offsets of the ground truth annotations of  $t$ . We compute the F1 score of system  $A_i$  with respect to the ground truth  $G$  for post  $t$  as mentioned in Equation 1 where  $|\cdot|$  denotes set cardinality.  $P^t$  and  $R^t$  measure the precision and recall, respectively.

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)} \quad (1)$$

Model	Trial F1	Test F1
MUDES RoBERTa	0.6886	0.6801
MUDES BERT	0.6771	0.6698
MUDES SPANBert	0.6751	0.6675
MUDES XLNet	0.6722	0.6653
BERT	0.6738	0.6538
BERT-CRF	0.6643	0.6517
BERT HateXplain	0.6387	0.6326
spaCy baseline	0.5976	0.5976
Bi-LSTM-CRF	0.5631	0.5398
Lexicon word match	0.3378	0.4086

Table 2: Results ordered by test F1 score. The Trial F1 column shows the F1 scores on the trial set and the Test F1 column shows the F1 scores for test set.

Observe in Table 2 that all of our deep neural-based models outperformed the spaCy baseline while the lexicon-based word match algorithm provided fairly good results despite it being an unsupervised method. Our best model is the MUDES RoBERTa model which scored 0.68 F1 score in the test set and is very compatible with the 0.70 F1 score that the best model scored in the competition. Furthermore, it is clear that the additional features supported by our MUDES framework, e.g., language modeling and ensembling, improves the results over a vanilla BERT transformer.

## 6 Conclusion and Future Work

In this paper, we presented the WLV-RIT approach for tackling the SemEval-2021 Task 5: Toxic Spans Detection. SemEval-2021 Task 5 provided participants with the opportunity of testing computational models to identify token spans in toxic posts as opposed to previous related SemEval tasks such as HatEval and OffensEval that provided participants with datasets annotated at the instance level. We believe that word-level predictions are an important step towards explainable offensive language identification.

We experimented with several methods including a lexicon-based word match, LSTMs, and neural transformers. Our results demonstrated that transformer models offered the best generalization results and, given the success observed, we developed MUDES, an open-source software framework based on neural transformers focused on detecting toxic spans in texts. With MUDES, we release two English models that performed best for this task (Ranasinghe and Zampieri, 2021a). A large model; en-large based on roberta-large which is more accurate, but has a low efficiency regarding space and time. The base model based on xlnet-base-cased; en-base is efficient, but has a comparatively low accuracy than the en-large model. All pre-trained models are available on Hugging Face Model Hub (Wolf et al., 2020)<sup>3</sup>. We also make MUDES available as a Python package<sup>4</sup> and set up as an open-source project<sup>5</sup>. In addition, a prototype User Interface (UI) of MUDES has been made accessible to the general public<sup>6</sup> based on Docker<sup>7</sup>.

In terms of future work, we would like to experiment with multi-task (neural) architectures that can be used for offensive language identification capable of carrying out predictions at both the word-level and post-level jointly. Furthermore, we would like to evaluate multi-task architectures on multi-domain and multilingual settings as well as broaden our experimental comparison to other types of recurrent network models, such as the Delta-RNN (Ororbia II et al., 2017).

<sup>3</sup>Available on <https://huggingface.co/mudes>

<sup>4</sup>Available at <https://pypi.org/project/mudes/>

<sup>5</sup>The MUDES GitHub repository is available at <https://github.com/tharindudr/MUDES>

<sup>6</sup>The UI can be accessed from <http://rgcl.wlv.ac.uk/mudes/>

<sup>7</sup>Available at <https://hub.docker.com/r/tharindudr/mudes>

## Acknowledgments

We would like to thank the shared task organizers for making this interesting dataset available. We further thank the anonymous SemEval reviewers for their insightful feedback.

## References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of WWW*.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019a. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of SemEval*.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019b. SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter. In *Proceedings of SemEval*.
- Çağrı Çöltekin. 2020. A Corpus of Turkish Offensive Language on Social Media. In *Proceedings of LREC*.
- Jonathan P Chang, Justin Cheng, and Cristian Danescu-Niculescu-Mizil. 2020. Don't let me be misunderstood: Comparing intentions and perceptions in online discussions. In *Proceedings of WWW*.
- Patricia Chiril, Farah Benamara Zitoune, Véronique Moriceau, Marlène Coulomb-Gully, and Abhishek Kumar. 2019. Multilingual and multitarget hate speech detection in tweets. In *Proceedings of TALN*.
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *Proceedings of ICWSM*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL*.
- Paula Fortuna, Joao Rocha da Silva, Leo Wanner, Sérgio Nunes, et al. 2019. A Hierarchically-labeled Portuguese Hate Speech Dataset. In *Proceedings of ALW*.
- Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of ALW*.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2019. Emoji powered capsule network to detect type and target of offensive posts in social media. In *Proceedings of RANLP*.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2020a. BRUMS at SemEval-2020 task 3: Contextualised embeddings for predicting the (graded) effect of context in word similarity. In *Proceedings of SemEval*.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2020b. InfoMiner at WNUT-2020 task 2: Transformer-based covid-19 informative tweet extraction. In *Proceedings of W-NUT*.
- Hansi Hettiarachchi and Tharindu Ranasinghe. 2021. TransWiC at SemEval-2021 Task 2: Transformer-based Multilingual and Cross-lingual Word-in-Context Disambiguation. In *Proceedings of SemEval*.
- Weipeng Huang, Xingyi Cheng, Taifeng Wang, and Wei Chu. 2019. BERT-Based Multi-Head Selection for Joint Entity-Relation Extraction. In *Proceedings of NLPCC*.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Vijayasaradhi Indurthi, Bakhtiyar Syed, Manish Shrivastava, Nikhil Chakravartula, Manish Gupta, and Vasudeva Varma. 2019. FERMI at SemEval-2019 task 5: Using sentence embeddings to identify hate speech against immigrants and women in Twitter. In *Proceedings of SemEval*.
- Tommi Jauhiainen, Tharindu Ranasinghe, and Marcos Zampieri. 2021. Comparing approaches to dravidian language identification. In *Proceedings of VarDial*.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Proceedings of TACL*.
- Ritesh Kumar, Atul Kr Ojha, Shervin Malmasi, and Marcos Zampieri. 2018. Benchmarking aggression identification in social media. In *Proceedings of TRAC*.
- Ritesh Kumar, Atul Kr. Ojha, Shervin Malmasi, and Marcos Zampieri. 2020. Evaluating Aggression Identification in Social Media. In *Proceedings of TRAC*.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *Proceedings of ICLR*.
- W. Li, S. Gao, H. Zhou, Z. Huang, K. Zhang, and W. Li. 2019. The Automatic Text Classification Method Based on BERT and Feature Union. In *Proceedings of ICPADS*.

- Ping Liu, Wen Li, and Liang Zou. 2019a. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of SemEval*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
- Jouni Luoma and Sampo Pyysalo. 2020. Exploring Cross-sentence Contexts for Named Entity Recognition with BERT. In *Proceedings of COLING*.
- Shervin Malmasi and Marcos Zampieri. 2017. Detecting Hate Speech in Social Media. In *Proceedings of RANLP*.
- Shervin Malmasi and Marcos Zampieri. 2018. Challenges in Discriminating Profanity from Hate Speech. *Journal of Experimental & Theoretical Artificial Intelligence*, 30:1–16.
- Thomas Mandl, Sandip Modha, Anand Kumar M, and Bharathi Raja Chakravarthi. 2020. Overview of the hasoc track at fire 2020: Hate speech and offensive language identification in tamil, malayalam, hindi, english and german. In *Proceedings of FIRE*.
- Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019a. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of FIRE*.
- Thomas Mandl, Sandip Modha, Prasenjit Majumder, Daksh Patel, Mohana Dave, Chintak Mandlia, and Aditya Patel. 2019b. Overview of the hasoc track at fire 2019: Hate speech and offensive content identification in indo-european languages. In *Proceedings of FIRE*.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of AACL*.
- Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhersch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of LREC*.
- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*.
- Hamdy Mubarak, Kareem Darwish, and Walid Magdy. 2017. Abusive language detection on Arabic social media. In *Proceedings of ALW*.
- Alexander G Ororbia II, Tomas Mikolov, and David Ritter. 2017. Learning simpler language models with the differential state framework. *Neural computation*, 29(12):3327–3352.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of SemEval*.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of EMNLP*.
- Zeses Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive Language Identification in Greek. In *Proceedings of LREC*.
- Tharindu Ranasinghe, Sarthak Gupte, Marcos Zampieri, and Ifeoma Nwogu. 2020. WLV-RIT at HASOC-Dravidian-CodeMix-FIRE2020: Offensive Language Identification in Code-switched YouTube Comments. In *Proceedings of FIRE*.
- Tharindu Ranasinghe and Hansi Hettiarachchi. 2020. BRUMS at SemEval-2020 task 12: Transformer based multilingual offensive language identification in social media. In *Proceedings of SemEval*.
- Tharindu Ranasinghe and Marcos Zampieri. 2020. Multilingual Offensive Language Identification with Cross-lingual Embeddings. In *Proceedings of EMNLP*.
- Tharindu Ranasinghe and Marcos Zampieri. 2021a. MUDES: Multilingual Detection of Offensive Spans. In *Proceedings of NAACL*.
- Tharindu Ranasinghe and Marcos Zampieri. 2021b. Multilingual Offensive Language Identification for Low-resource Languages. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*.
- Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. 2019. BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification. In *Proceedings of FIRE*.
- Michael Ridenhour, Arunkumar Bagavathi, Elaheh Raisi, and Siddharth Krishnan. 2020. Detecting Online Hate Speech: Approaches Using Weak Supervision and Network Embedding Models. In *Proceedings of SBP-BRIMS*.
- Hugo Rosa, N Pereira, Ricardo Ribeiro, Paula Costa Ferreira, Joao Paulo Carvalho, S Oliveira, Luísa Coheur, Paula Paulino, AM Veiga Simão, and Isabel Trancoso. 2019. Automatic cyberbullying detection: A systematic review. *Computers in Human Behavior*, 93:333–345.
- Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A large-scale semi-supervised dataset for offensive language identification. *arXiv preprint arXiv:2004.14454*.

- Gudbjartur Ingi Sigurbergsson and Leon Derczynski. 2020. Offensive Language and Hate Speech Detection for Danish. In *Proceedings of LREC*.
- Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. 2020. Portuguese Named Entity Recognition using BERT-CRF. *arXiv preprint arXiv:1909.10649*.
- Julia Maria Struß, Melanie Siegel, Josef Ruppenhofer, Michael Wiegand, Leibniz ScienceCampus, and Manfred Klenner. 2019. Overview of germeval task 2, 2019 shared task on the identification of offensive language. In *Proceedings of KONVENS*.
- Stéphan Tulkens, Lisa Hilte, Elise Lodewyckx, Ben Verhoeven, and Walter Daelemans. 2016. A Dictionary-based Approach to Racism Detection in Dutch Social Media. In *Proceedings of TA-COS*.
- Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of NAACL Student Research Workshop*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding. In *Proceedings of NeurIPS*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of NAACL*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffenseEval). In *Proceedings of SemEval*.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffenseEval 2020). *Proceedings of SemEval*.
- Justine Zhang, Jonathan Chang, Cristian Danescu-Niculescu-Mizil, Lucas Dixon, Yiqing Hua, Dario Taraborelli, and Nithum Thain. 2018. Conversations gone awry: Detecting early signs of conversational failure. In *Proceedings of ACL*.

# YNU-HPCC at SemEval-2021 Task 5: Using a Transformer-based Model with Auxiliary Information for Toxic Span Detection

Ruijun Chen, Jin Wang and Xuejie Zhang  
School of Information Science and Engineering  
Yunnan University  
Kunming, China

Contact: chenrj@mail.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

## Abstract

Toxic span detection requires the detection of spans that make a text toxic instead of simply classifying the text. In this paper, a transformer-based model with auxiliary information is proposed for SemEval-2021 Task 5. The proposed model was implemented based on the BERT-CRF architecture. It consists of three parts: a transformer-based model that can obtain the token representation, an auxiliary information module that combines features from different layers, and an output layer used for the classification. Various BERT-based models, such as BERT, ALBERT, RoBERTa, and XLNET, were used to learn contextual representations. The predictions of these models were assembled to improve the sequence labeling tasks by using a voting strategy. Experimental results showed that the introduced auxiliary information can improve the performance of toxic spans detection. The proposed model ranked 5th of 91 in the competition. The code of this study is available at [https://github.com/Chenrj233/semEval2021\\_task5](https://github.com/Chenrj233/semEval2021_task5)

## 1 Introduction

Existing toxicity detection datasets and models classify the entire comment or document and do not identify the range that makes the text toxic. A system that accurately locates the toxicity range in the text is crucial in achieving semi-automatic review. As a complete submission for the shared task, systems are required to extract a list of toxic spans or an empty list per text. We define a sequence of words that attribute to the text’s toxicity as the toxic span. Table 1 shows two toxic spans, ”stupid” and ”a!@#!@,” which have character offsets from 10 to 15 (counting starts from 0) and from 51 to 56, respectively. Systems are then expected to return the offset list for this text.

<b>Text</b>
This is a stupid example, so thank you for nothing a!@#!@.
<b>Offset List</b>
[10,11,12,13,14,15,51,52,53,54,55,56]

Table 1: Example of toxic spans detection shared task.

The main purpose of this task is to identify the toxic spans in a given text; this can be transformed into a sequence labeling task in natural language processing. Unlike normal sequence labeling tasks, this task is more challenging because the toxic spans in the text may involve a word, phrase, or even a sentence. Traditional methods used to address the problem of sequence labeling include conditional random fields (CRF) (Lafferty et al., 1999), combined models of both long-short-term memory and CRF (LSTM-CRF) (Gupta et al., 2019), and bidirectional encoder representation from transformers (BERT) (Devlin et al., 2019).

In this study, we use BERT, ALBERT (Lan et al., 2019), RoBERTa (Liu et al., 2019), and XLNET (Yang et al., 2019) to solve this problem. Compared with the conventional model, our model adds auxiliary information to improve the performance in this task. After a simple analysis of the text data, it can be found that not all the words in the toxic span have a toxic meaning, and some toxic meanings occur in a specific context or semantic conditions. Therefore, if the tokens can be classified with the auxiliary information such as sentence representation, the performance of the model will improve. The results of the experiment prove that some of the proposed methods are effective. By using ensemble learning, we merge the results of the BERT, ALBERT, RoBERTa, and XLNET models into the final prediction, obtaining the 5th rank out of 91 and a  $F_1$  score of 0.696.

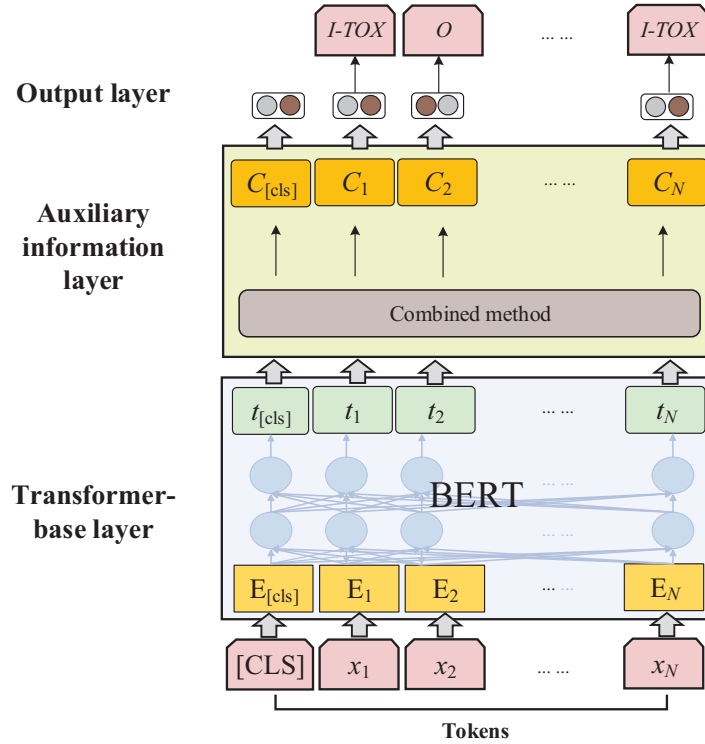


Figure 1: Overall architecture of the proposed transformer-based model with auxiliary information.

The remainder of this paper is organized as follows. Section 2 describes the specific structure of the adopted model. The experimental results are summarized in Section 3. Finally, Section 4 presents the conclusions of the study.

## 2 Transformer-based Model with Auxiliary Information

Figure 1 shows the architecture of the proposed model, which consists of three layers: a transformer-based layer, an auxiliary information layer, and an output layer. The transformer-based layer can be BERT, ALBERT, RoBERTa, XLNET, or any other transformer-based model. In the auxiliary information layer, several approaches are applied to combine token representation. The combined token representations are used in the output layer to output the label of each token.

### 2.1 Transformer-based Layer

The transformer-based layer is the first part of the model. The purpose of this layer is to obtain the representation of tokens and the entire text. For illustration, we can use the BERT-large (Devlin et al., 2019) model to produce token representations from each layer. With BERT-large, 25 layers of token representation vectors can be obtained: one embed-

ding representation and twenty-four hidden states. Unlike previous methods, 25 layers of token representation vectors are combined by using several methods in the next layer. The representations produced by the transformer-based layer are then fed into the next layer.

### 2.2 Auxiliary Information Layer

The traditional method directly passes the token representation vectors to the classification layer. To improve the performance of the model, we attempt to combine token representation vectors and the sentence representation vector in different ways. Figure 2 depicts the attempted methods, which are described as follows:

- **Method 1.** Token vector of the last layer and the sentence vector.
- **Method 2.** Token vector of the last layer concatenated with the sentence vector.
- **Method 3.** Linear combination of the token vector of each layer.
- **Method 4.** Linear combination of the token vector of each layer and the sentence vector.

The combined representation of tokens passes on to the next layer.

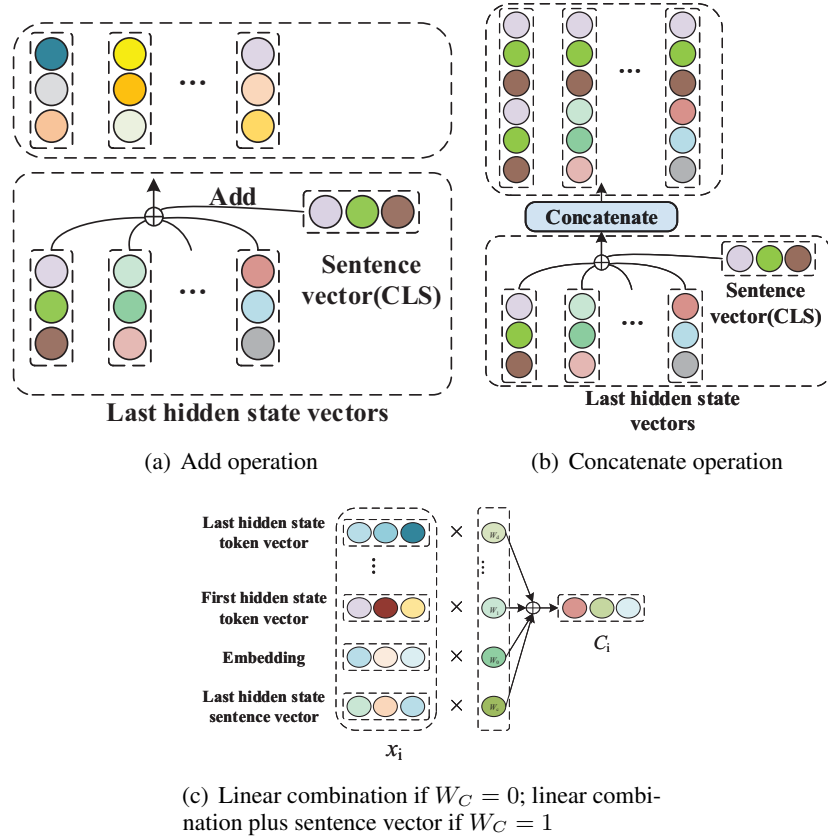


Figure 2: Different types of representations in auxiliary information layer.

### 2.3 Output Layer

The output layer is a fully connected dense layer with softmax activation. It aims to classify whether a token belongs to the toxic span in a text. The combined representation of each token passed by the auxiliary information layer is the input of this layer, and the output layer predicts the labels for the candidate tokens. The loss function of the proposed model is the categorical cross-entropy.

## 3 Experimental Results

In this section, we present the comparative results of the proposed model.

### 3.1 Dataset

During the competition, we used only the data (Pavlopoulos et al., 2021) provided by the task organizer for the experiments. This task involves trial data (which include 689 posts and spans), training data (which include 7939 posts and spans), and test data (which include 2000 posts). We used the training data as the training set and trial data as the validation set. We needed to find the subscript offset set of the toxic spans of each post in the test

data.

As this is a sequence labeling task, a common data preprocessing method is to use the BIO tagging format. We observed better performance when the IO tagging format was adopted during the actual training process. Therefore, our output layer was a two-classification layer that outputs the probability of a token belonging to a toxic span.

### 3.2 Evaluation Metrics

For this task, we employed the  $F_1$ -score metric (da San Martino et al., 2020) to evaluate the responses of a system participating in the challenge.

For each post,  $t_i$ , the predicted span was a set,  $S_i$ , of character offsets and  $G_i$  was the character offset of the groundtruth annotations of  $t_i$ . The  $F_1$  score of  $t_i$  was calculated as follows:

$$F_1(S_i, G_i) = \frac{2 * P(S_i, G_i) * R(S_i, G_i)}{P(S_i, G_i) + R(S_i, G_i)} \quad (1)$$

where  $P(S_i, G_i)$  and  $R(S_i, G_i)$  are respectively precision and recall scores defined as follows:

$$P(S_i, G_i) = \frac{|S_i \cap G_i|}{|S_i|} \quad (2)$$

Transformer model	Auxiliary information	Validation set	Test set
<b>BERT-large</b>	None	0.649	0.679
	Add sentence vector	0.670	0.679
	Concatenate sentence vector	0.671	0.671
	Linear combination	0.661	0.683
	Linear combination plus sentence vector	0.672	0.679
<b>ALBERT-xlarge</b>	None	0.659	0.675
	Add sentence vector	0.665	0.665
	Concatenate sentence vector	0.656	0.668
	Linear combination	0.648	0.667
	Linear combination plus sentence vector	0.657	0.670
<b>RoBERTa-large</b>	None	0.656	0.676
	Add sentence vector	0.620	0.662
	Concatenate sentence vector	0.610	0.673
	Linear combination	0.663	0.667
	Linear combination plus sentence vector	0.667	0.667
<b>XLNET-large</b>	None	0.659	0.679
	Add sentence vector	0.674	0.674
	Concatenate sentence vector	0.669	0.669
	Linear combination	0.674	0.675
	Linear combination plus sentence vector	0.678	0.681

Table 2:  $F_1$ -score of different models on validation set and test set.

$$R(S_i, G_i) = \frac{|S_i \cap G_i|}{|G_i|} \quad (3)$$

If  $G_i$  is empty for some post  $t_i$ , we set  $F_1(S_i, G_i) = 1$  if  $S_i$  is also empty and  $F_1(S_i, G_i) = 0$  otherwise. Finally, we averaged  $F_1(S_i, G_i)$  over all posts  $t_i$ .

### 3.3 Implementation Details

Each model was fine-tuned for eight epochs. We used the Adam (Kingma and Ba, 2015), AdamW (Loshchilov and Hutter, 2017), and Stochastic Gradient Descent (SGD) algorithm for optimization. The final one used was AdamW with a learning rate of  $5e - 6$ .

In the training process, we attempted to use the cross-entropy loss, focal loss (Lin et al., 2020), and Dice loss (Li et al., 2020). The results on the validation set showed that the focal loss and Dice loss are better than the cross-entropy loss. This may be due to an imbalance between the toxic and nontoxic categories in the text. In order to compare with the baseline model, we finally used the cross-entropy loss function to train all models.

### 3.4 Comparative Results

We used BERT, ALBERT, RoBERTa, and XLNET as the transformer-based layers. The model exhibit-

ing the best performance on the validation set in the eight epochs was used to predict the spans on the test set in the competition. The results on the test set are presented in Table 2. The model that performed the best on the test set over the eight epochs is also presented in Table 2.

In terms of the performance on the validation set, the BERT and XLNET models with the auxiliary information layer are better than those without. Method 4, mentioned earlier, achieves the highest  $F_1$  score. In case of ALBERT, only method 1 improves the performance. Methods 3 and 4 can improve the performance of RoBERTa.

Regardless of the performance on the validation set, the  $F_1$  score increases by 0.004 when using method 3 in the BERT model and increases by 0.002 when using method 4 in XLNET. The auxiliary information layer does not improve the performance of ALBERT and RoBERTa.

The results show that the performance of the best-performing model on the validation set is significantly different from that of the best-performing model on the test set. The reason for this difference may be the inconsistent data distribution of the validation and test sets.

However, the results indicate that when the validation set is not appropriate, the auxiliary informa-



tion layer can effectively improve the performance of the baseline model on the validation set. The BERT and XLNET models are the most suitable for the auxiliary information layer.

## 4 Conclusion

In this paper, we introduce the method we used in SemEval-2021 Task 5. We improved the performance of the basic model by reducing the number of categories for each token, selecting the appropriate loss function, adding some additional information to the representation vector of the tokens during classification, and finally obtaining a model that can detect the toxicity in a text. Our experimental results showed that adding auxiliary information to the original token representation vector is helpful in sequence labeling tasks.

In addition, we found that the model has some limitations. After analyzing the prediction results, we observed that although the model can learn the representation of each token well, token classification errors can occur when some tokens are toxic without the entire text being toxic. One possible solution for this is to add a text classification task to train the model.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61702443, 61966038 and 61762091.

## References

- Giovanni da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2020. [Fine-grained analysis of propaganda in news articles](#). *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, (January):5636–5646.
- Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1(Mlm):4171–4186.
- Pankaj Gupta, Khushbu Saxena, Usama Yaseen, Thomas Runkler, and Hinrich Schütze. 2019. [Neural architectures for fine-grained propaganda detection in news](#). *arXiv*.
- Diederik P. Kingma and Jimmy Lei Ba. 2015. [Adam: A method for stochastic optimization](#). *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 1999. [Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data](#). *Abstract*. 2001(June):282–289.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [Albert: A lite bert for self-supervised learning of language representations](#). *arXiv*, pages 1–17.
- Xiaoya Li, Xiaofei Sun, Yuxian Meng, Junjun Liang, Fei Wu, and Jiwei Li. 2020. [Dice Loss for Data-imbalanced NLP Tasks](#). pages 465–476.
- Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. 2020. [Focal Loss for Dense Object Detection](#). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *arXiv*, (1).
- Ilya Loshchilov and Frank Hutter. 2017. [Decoupled weight decay regularization](#). *arXiv*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [XLNet: Generalized autoregressive pretraining for language understanding](#). *arXiv*, (NeurIPS):1–18.

# UIT-ISE-NLP at SemEval-2021 Task 5: Toxic Spans Detection with BiLSTM-CRF and ToxicBERT Comment Classification

**Son T. Luu**

University of Information Technology  
Vietnam National University  
Ho Chi Minh City, Vietnam  
sonlt@uit.edu.vn

**Ngan Nguyen**

University of Information Technology  
Vietnam National University  
Ho Chi Minh City, Vietnam  
ngannlt@uit.edu.vn

## Abstract

We present our works on SemEval-2021 Task 5 about Toxic Spans Detection. This task aims to build a model for identifying toxic words in whole posts. We use the BiLSTM-CRF model combining with ToxicBERT Classification to train the detection model for identifying toxic words in posts. Our model achieves 62.23% by F1-score on the Toxic Spans Detection task.

## 1 Introduction

Detecting toxic posts on social network sites is a crucial task for social media moderators in order to keep a clean and friendly space for online discussion. To identify whether a comment or post is toxic or not, social network administrators often read the whole comment or post. However, with a large number of lengthy posts, the administrators need assistance to locate toxic words in each post to decide whether a post is toxic or non-toxic instead of reading the whole post. The SemEval-2021 Task 5 (Pavlopoulos et al., 2021) provides a valuable dataset called Toxic Spans Detection dataset in order to train the model for detecting toxic words in lengthy posts.

Based on the dataset from the shared task, we implement the machine learning model for detecting toxic words posts. Our model includes: the BiLSTM-CRF model (Lample et al., 2016) for detecting the toxic spans in the post, and the ToxicBERT (Hanu and Unitary team, 2020) for classifying whether the post is toxic or not. Before training the model, we pre-process texts in posts and encode them by the GloVe word embedding (Pennington et al., 2014). Our model achieves 62.23% on the test set provided by the task organizers.

## 2 Related works

Many corpora are constructed for toxic speech detection problems. They consist of flat label and

hierarchical label datasets. The flat label datasets only classify one label for each comment in the dataset (e.g., hate, offensive, clean), while hierarchical datasets can classify multiple aspects of the comment (e.g., hate about racism, hate about sexual oriented, hate about religion, and hate about disability). For flat label, we present several datasets including the two datasets which are provided by Waseem and Hovy (2016) and Davidson et al. (2017) in English, the dataset of Albadi et al. (2018) in Arabic, and the dataset of by Alfina et al. (2017) in Indonesian. For the hierarchical label, we introduce the dataset constructed by Zampieri et al. (2019) in English, the dataset provided by Fortuna et al. (2019) in Portuguese, and the CO-NAN dataset by Chung et al. (2019), which is the multilingual corpus (constructed in Italian, English, and French).

In addition, many of shared tasks about hate speech and abusive languages are organized, such as the SemEval-2019 Task 5 (Multilingual) (Basile et al., 2019), the SemEval-2019 Task 6 (English) (Zampieri et al., 2019), the PolEval 2019 Shared task 6 (Polish) (Ptaszynski et al., 2019), the GermEval 2018 (Germany) (Wiegand et al., 2018), EVALITA 2019 (Italian) (Bosco et al., 2018), Toxic Comment Classification Challenge<sup>1</sup>, and VLSP 2019 Shared task (Vietnamese) (Vu et al., 2019).

Besides, SOTA approaches like deep learning (Badjatiya et al., 2017) and transformers models (Isaksen and Gambäck, 2020) are applied in the hate speech detection and toxic posts classification. However, these models only classify based on the whole posts or documents. For the Toxic Spans Detection task, we adapt the mechanism from Sequence tagging (Wang et al., 2020) and Name entities Recognition (Lin et al., 2017) for detecting toxic words from posts.

<sup>1</sup><https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

### 3 Dataset

The dataset is provided from the SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021). It includes the training and the test sets. Both of them consist of two parts: the content of posts and the spans denoting the toxic words in the posts. Spans represent toxic words in the posts as a set of character indexes. Table 1 illustrates several examples from the training set.

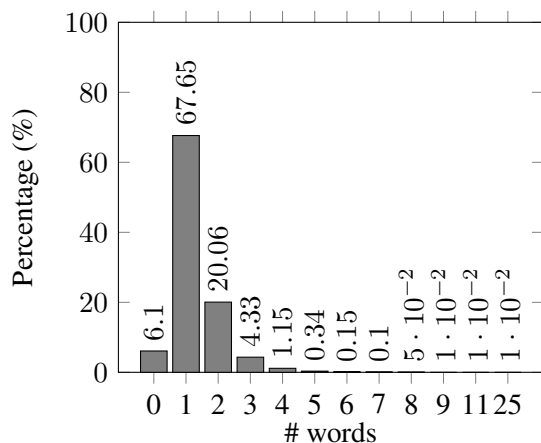


Figure 1: Number of toxic words in spans for each post in the training set.

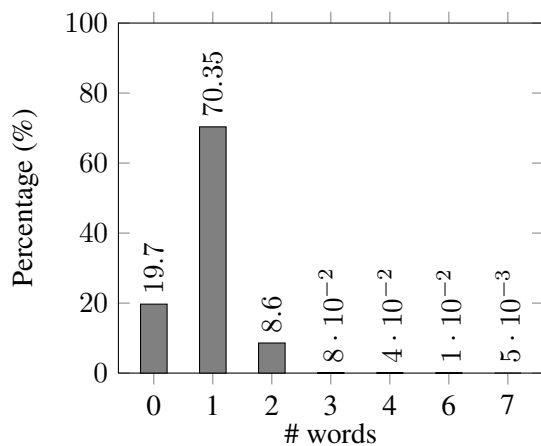


Figure 2: Number of toxic words in spans for each post in the test set.

According to Table 1, a post contains multiple spans of toxic words. For each span, it contains a single word, a phrase, or a sentence. As described in Figure 1, most of the spans in the training set are single words, which account for 67.65%, while only 20.06% of spans contains two words, and 6.1% of spans is empty. Posts whose spans contain more than two words in the dataset are few. Espe-

cially in the training set, there is a post in which spans contain 25 words.

Besides, Figure 2 illustrates the number of toxic words in spans per post for the test set. Spans containing single words account for the highest percentage (70.35%) in the test set, and are higher than in the training set, while the multiple-word spans are few. Also, the empty spans in the test set are higher than the training set, and the longest post in the test set contains only seven words.

### 4 System description

#### 4.1 Data preparation

With the given dataset from the SemEval-2021 Task 5 about Toxic Spans Detection (Pavlopoulos et al., 2021), we firstly transform spans into a set of words. Then, we pre-process the posts as follows: (1) Segmenting the posts by the TweetTokenizer from nltk<sup>2</sup>, and (2) Changing texts to lower case.

#### 4.2 Feature extraction

We use the glove.twitter.27b.25d word embedding<sup>3</sup> to construct the dictionary and encode the text of posts. Posts are encoded by the dictionary of the word embedding. The  $\langle UNK \rangle$  tokens are added if a word in posts is not found in the dictionary. To make sure all vectors are the same length, we add the  $\langle PAD \rangle$  token. Then, we set the maximum length of vectors equal to 128. Spans are transformed into a one-hot vector corresponding to each word in posts where toxic words are denoted as 1 and others are denoted as 0. Table 2 illustrates an example of encoding data in our system.

#### 4.3 Training models

**Detection model:** BiLSTM-CRF is a deep neural model used for Named-entity recognition task (Lample et al., 2016). We implement this model for the task of detecting toxic words in documents. The model includes three main layers: (1) The word representation layer uses embedding matrix from the GloVe word embedding, (2) The BiLSTM layer for sequence labeling, and (3) The Conditional Random Field (CRF) layer to control the probability of output labels. The output is a binary vector, in which each value determines whether

<sup>2</sup><https://www.nltk.org/api/nltk.tokenize.html>

<sup>3</sup><https://nlp.stanford.edu/projects/glove/>

No	Posts	Spans
1	What a <b>knucklehead</b> . How can anyone not know this would be offensive??	[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17]
2	I only use the word haole when <b>stupidity</b> and <b>arrogance</b> is involved and not all the time. Excluding the POTUS of course.	[31, 32, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 50, 51, 52, 53]
3	<b>Such garbage logic by republicans</b> which will backfire and rush america into the great depression II	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32]
4	what a <b>hypocrite</b> of bs,, tell us loser how you live without gasoline, plastic, medical needs and medications, all from OIL,, but you cant of course so you <b>ignorant</b> fools in your <b>hypocrisy</b> spew this bs	[7, 8, 9, 10, 11, 12, 13, 14, 15, 155, 156, 157, 158, 159, 160, 161, 162, 178, 179, 180, 181, 182, 183, 184, 185, 186]
5	Exposing hypocrites like Trump and Pence is therapeutic for you? Good job!	[]

Table 1: Sample posts from the training set. The toxic span are highlighted as bold.

	Original	Transformed
Text	I only use the word haole when <b>stupidity</b> and <b>arrogance</b> is involved and not all the time. Excluding the POTUS of course.	['i', 'only', 'use', 'the', 'word', 'haole', ...] <b>Vector:</b> [12, 216, 718, 15, 894,...]
Spans	[31, 32, 33, 34, 35, 36, 37, 38, 39, 45, 46, 47, 48, 49, 50, 51, 52, 53]	['i', 'only', 'use', 'the', 'word', 'haole', 'when', ' <b>stupidity</b> ', 'and', ' <b>arrogance</b> ', 'is', ...] <b>Vector:</b> [0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0 ...]

Table 2: Example of encoding data into vectors.

a word is toxic or non-toxic. The architecture of BiLSTM-CRF is described in Figure 3.

**Classification model:** The ToxicBERT model (Detoxify) is introduced by Hanu and Unitary team (2020) with the purpose to stop online abusive comments. It is a pre-trained model and is easy to use by using transformers library<sup>4</sup>. The model is trained on the Toxic Comments Classification Challenge datasets provided by Jigsaw.

Our system combines the detection and classification model together. The detection model (BiLSTM-CRF) returns the toxic spans from the post, while the classification model (ToxicBERT) classifies whether a post is toxic or non-toxic. If a post is non-toxic, the classification model returns an empty span. By contrast, it reserves the spans of the detection model. Then, predicted spans are decoded to character indexes for submission. Our system is illustrated in Figure 4

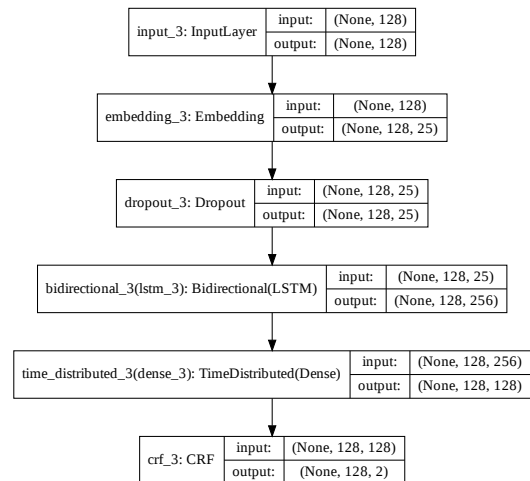


Figure 3: The BiLSTM-CRF model architecture.

<sup>4</sup><https://huggingface.co/unitary/toxic-bert>

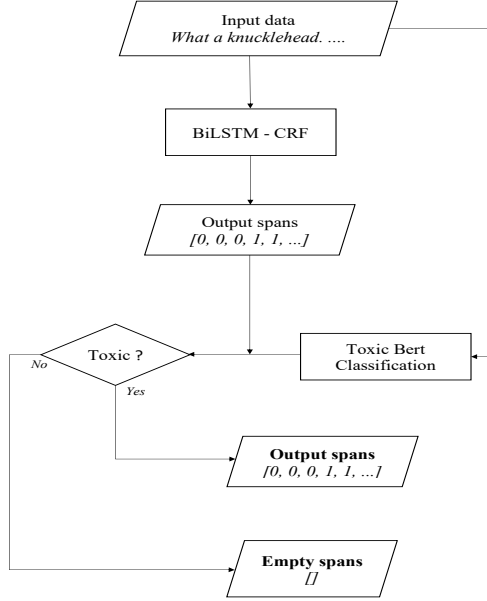


Figure 4: Our system architecture.

## 5 Experimental results

### 5.1 Evaluation metric

The variant version of F1-score is used to evaluate the results of the competition (Da San Martino et al., 2019). Let  $T$  is the total of post in the dataset,  $T = [t_1, t_2, \dots, t_n]$ ,  $n$  is the number of posts,  $A$  is spans given by the model, and  $G$  is ground truth spans.

The F1-score over the dataset is defined as:

$$\frac{1}{|T|} \sum_t F_1^t = 2 * \frac{P^t(A, G) * R^t(A, G)}{P^t(A, G) + R^t(A, G)} \quad (1)$$

In the Equation 1,  $P^t$  determines the precision, and  $R^t$  determines the recall of the post  $t$ . The precision and recall are calculated as Equation 2 and Equation 3, respectively. The  $S^t$  in both Equation 2 and Equation 3 is set of toxic characters of post  $t$  (span).

$$P^t(A, G) = \frac{|S_A^t \cap S_G^t|}{S_A^t} \quad (2)$$

$$R^t(A, G) = \frac{|S_A^t \cap S_G^t|}{S_G^t} \quad (3)$$

### 5.2 Main result from task

Model	Result (%)
BiLSTM-CRF	61.32
<b>BiLSTM-CRF + ToxicBERT</b>	<b>62.23</b>
Top 1 rank	<b>70.83</b>

Table 3: Experimental results obtained by our system in comparison with Top 1 rank result.

According to Table 3, when only BiLSTM-CRF is used, the result by F1 score is 61.32%. The result increases up to 62.23% when we applied ToxicBERT Classifier, and this is our final result of the shared task (ranked 63<sup>th</sup> among 92 teams).

### 5.3 Error analysis

According to Table 4, the appearance of the ToxicBERT classifier can make a better prediction for the non-toxic posts (See example No. 1). This increases the performance of our system, however, not significantly, because the number of empty toxic span comments in the test set is not too much (as described in Section 3). Apart from empty spans, wrong predictions are fell into the case as example No.3. Although the ToxicBERT model predicts this example as non-toxic, the BiLSTM-CRF model predicts it as empty spans, and thus the result is empty spans according to the result of the BiLSTM-CRF model. For example No.2, the system returns wrong spans for the first span in the post. Finally, our system cannot predict well for spans that contain more than two words. It returns spans, but not enough, as shown in examples No.4 and No.5 from Table 4.

## 6 Conclusion

We use the BiLSTM-CRF and ToxicBERT models for detecting toxic words in the posts. Our model achieves 62.23% by F1-score from the competition. From the error analysis, we found that our model predicts well just for single-word spans and empty spans.

In further researches, we improve the performance of the detection model by applying the attention mechanism and using the character-level representation combining with word-level representation. Character-level models like CharBERT (Ma et al., 2020) is a potential approach to increase the performance of toxic spans detection tasks.

No.	Posts	Gold span	Prediction span: BiLSTM-CRF	Prediction span: BiLSTM-CRF+ ToxicBERT
1	Indeed, people the world over, all know that President Trump is a loser!	[]	[66, 67, 68, 69, 70]	[]
2	Idiot "miner" in the photo isn't even wearing a dust mask. MSHA needs to inspect this operation for more worker safety violations.	[0, 1, 2, 3, 4]	[-1, 0, 1, 2, 3]	[-1, 0, 1, 2, 3]
3	People who are anti-immigration are weak, lying, racist whiners.	[49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	[]	[]
4	Uh-No, keep voting for failed Liberal idiocy that guarantees results ala Detroit, Chicago, etc. You'll wish your body had only some crap rather than gang-banger gunfire.	[38, 39, 40, 41, 42, 43]	[38, 39, 40, 41, 42, 43, 133, 134, 135, 136]	[38, 39, 40, 41, 42, 43, 133, 134, 135, 136]
5	What is he going to do about those toxic mercury florescent bulbs Bush and Gore pushed on the stupid American public?	[94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115]	[94, 95, 96, 97, 98, 99]	[94, 95, 96, 97, 98, 99]

Table 4: Several wrong predictions on the test set by our system.

## References

- N. Albadi, M. Kurdi, and S. Mishra. 2018. Are they our brothers? analysis and detection of religious hate speech in the arabic twittersphere. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 69–76.
- I. Alfina, R. Mulia, M. I. Fanany, and Y. Ekanata. 2017. Hate speech detection in the indonesian language: A dataset and preliminary study. In *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 233–238.
- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. *Proceedings of the 26th International Conference on World Wide Web Companion - WWW '17 Companion*.
- Valerio Basile, Cristina Bosco, Elisabetta Fersini, Debora Nozza, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, and Manuela Sanguinetti. 2019. *SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter*. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 54–63, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Cristina Bosco, Dell'Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.
- Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. *CONAN - COUNTER NARRATIVES THROUGH NICHE SOURCING: A MULTILINGUAL DATASET OF RESPONSES TO FIGHT ONLINE HATE SPEECH*. In *Proceedings of the 57th Annual Meet-*

- ing of the Association for Computational Linguistics, pages 2819–2829, Florence, Italy. Association for Computational Linguistics.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.
- Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.
- Paula Fortuna, João Rocha da Silva, Juan Soler-Company, Leo Wanner, and Sérgio Nunes. 2019. A hierarchically-labeled Portuguese hate speech dataset. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 94–104, Florence, Italy. Association for Computational Linguistics.
- Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.
- Vebjørn Isaksen and Björn Gambäck. 2020. Using transfer-based language models to detect hateful and offensive language online. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, Online. Association for Computational Linguistics.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.
- Bill Y. Lin, Frank Xu, Zhiyi Luo, and Kenny Zhu. 2017. [Multi-channel BiLSTM-CRF model for emerging named entity recognition in social media](#). In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 160–165, Copenhagen, Denmark. Association for Computational Linguistics.
- Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. [CharBERT: Character-aware pre-trained language model](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Michał Ptaszynski, Agata Pieciukiewicz, and Paweł Dybała. 2019. Results of the poleval 2019 shared task 6: First dataset and open shared task for automatic cyberbullying detection in polish twitter.
- Xuan-Son Vu, Thanh Vu, Mai-Vu Tran, Thanh Le-Cong, and Huyen T M. Nguyen. 2019. HSD shared task in VLSP campaign 2019: Hate speech detection for social good. In *Proceedings of VLSP 2019*.
- Yan Wang, Jiayu Zhang, Jun Ma, Shaojun Wang, and Jing Xiao. 2020. [Contextualized emotion recognition in conversation as sequence tagging](#). In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 186–195, 1st virtual meeting. Association for Computational Linguistics.
- Zeeraq Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93, San Diego, California. Association for Computational Linguistics.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota.

# GHOST at SemEval-2021 Task 5: Is explanation all you need?

**Kamil Pluciński** and **Hanna Klimczak**

Institute of Computer Science

Poznan University of Technology, 60-965 Poznań, Poland

{kamil.plucinski97, hanna.klimczak}@gmail.com

## Abstract

This paper discusses different approaches to the Toxic Spans Detection task. The problem posed by the task was to determine which words contribute mostly to recognising a document as toxic. As opposed to binary classification of entire texts, word-level assessment could be of great use during comment moderation, also allowing for a more in-depth comprehension of the model's predictions. As the main goal was to ensure transparency and understanding, this paper focuses on the current state-of-the-art approaches based on the explainable AI concepts and compares them to a supervised learning solution with word-level labels. The work consists of two xAI approaches that automatically provide the explanation for models trained for binary classification of toxic documents: an LSTM model with attention as a model-specific approach and the Shapley values for interpreting BERT predictions as a model-agnostic method. The competing approach considers this problem as supervised token classification, where models like BERT and its modifications were tested. The paper aims to explore, compare and assess the quality of predictions for different methods on the task. The advantages of each approach and further research direction are also discussed.

## 1 Introduction

The popularity of social media platforms has been continuously increasing over time. As reported by [Social \(2020\)](#), 3.8 billion people have been active users of social media in January 2020. While user inter-connectivity carries a lot of positive effects, there is still a significant threat of cyberbullying and harassment caused by the illusion of anonymity online. Statistics released by Facebook ([Richter, 2020](#)) identified that in the first quarter of 2020, there were 2.3 million bullying/harassment posts

and 9.6 million hate speech posts detected as a violation of Community Standards.

The importance of keeping the online community safe for users has caused many researchers to focus their work on detecting toxic contents in order to assist moderators in their difficult and mentally exhausting work. A lot of progress has been done in the task of toxic comment classification, but unfortunately, complex models still lack clear explanation and cannot gain much moderators' trust. A step towards increasing the transparency and therefore trust in automatic comment classifiers would be extending current approaches to operate on word-level rather than document-level. The ability to extract and highlight key text fragments that cause the toxic character of a comment could be of great use in post moderation.

The lack of extensive datasets for word-level toxicity detection poses an obstacle to traditional, supervised learning classification, as current state-of-the-art models are very complex and normally require large-scale data. Therefore, with the subject of this task being defined in terms of understanding and transparency for endpoint users, the authors decided to explore the explainable AI methodology. Ongoing research in xAI community examines many different approaches, with the two of them being in the focus of current work: model-specific attention-based ([Mohankumar et al., 2020](#)) and model-agnostic ([Lundberg and Lee, 2017](#)) explanations. This paper aims to compare the results obtained by these methods on Toxic Spans Detection task ([Pavlopoulos et al., 2021](#)) for the English language using supervised models ([Devlin et al., 2018](#)), acting as a baseline approach.

This paper is organised as follows. Related works and the backgrounds of discussed methods are described in Section 2. Section 3 presents three approaches to toxic spans detection, each with method-specific details in the corresponding



subsection. Finally, the results of the experiments are included in Section 4, with the discussion and conclusions in Section 5 and 6 respectively.

## 2 Related work

Pavlopoulos et al. (2017) introduced the application of a GRU-based Recurrent Neural Network for toxicity detection in documents. This approach was compared to the previous state-of-the-art for this task, presented in Wulczyn et al. (2017), which involved the use of Logistic Regression and Multi-layer Perceptron and operated on n-grams. Another model compared in their work was a Convolutional Neural Network with pretrained word embeddings. RNN model outperformed other approaches, and it was further extended by the attention module, which improved the quality of classification.

Attention has been commonly used by many researchers as a medium of explanation for the predictions made by the model (Ghaeini et al., 2018). It allows to analyse what part of the input the model focuses on the most while making a prediction. Current research does not provide a unified answer to whether the attention mechanism in models is a good source of explanation. Experiments carried out in Jain and Wallace (2019) point out that attention might not be a reliable method for interpreting the model's predictions. This was due to the fact that the attention distribution did not align well with other feature-importance measures. The other argument was that providing very different attention distribution often does not impact the predictions made by the model significantly. However, Wiegrefe and Pinter (2019) challenges the aforementioned work, claiming that while the answers given by attention should not be uncontrollably trusted and the definition of explanation must be clearly stated, it could still be a useful tool for model understanding and should not be disregarded easily.

An alternative formulation of the Toxic Spans Detection is to treat it as a supervised learning task, which involves training the models to predict the toxicity of each word separately. Previously used, Recurrent Neural Networks have been replaced as the state-of-the-art approach for many tasks by transformer-based models (Vaswani et al., 2017). Transformer architecture, consisting of an encoder and decoder enriched with multi-head attention modules, turned out to outperform previous approaches on a series of NLP tasks. Bidirectional

Encoder Representations from Transformers (Devlin et al., 2018) has been proposed as a powerful tool for many language-based problems. Typically, BERT is pre-trained on two unsupervised tasks when it is fed with large-scale text corpora and later adapted to a specific task during the fine-tuning stage, requiring much less data and computing time.

## 3 Toxic Spans Detection

This section describes the following approaches: analysing the attention of an LSTM model with orthogonalization of hidden states 3.1; using SHAP to provide explanation of BERT predictions for toxic comment classification 3.2; training BERT for classification of toxic tokens 3.3.

### 3.1 Orthogonal LSTM

As stated in the previous section, attention in RNN models is still a questionable medium of explanation. An interesting approach to LSTM with attention has been proposed in Mohankumar et al. (2020). The authors claim, that attention vectors in LSTM models are too similar to each other and therefore, cannot be used to explain the predictions. They propose an orthogonalization technique to increase the conicity of hidden states, enabling better interpretability. The equations of LSTM units are updated in order to orthogonalize the hidden state at a given time with respect to the previous states. The implementation used in this work has been adapted from the source code<sup>1</sup> provided by the authors of the original paper (Mohankumar et al., 2020).

During the preprocessing stage, the inputs with toxicity higher than 0.5 were considered as toxic examples. Due to the memory limitations of our computing architecture, the entire CivilComments (Borkan et al., 2019) dataset could not be used for training. Therefore, Random Undersampling (Cochran, 1977) of majority class was performed. The resulting set consisted of 350000 examples. The text was tokenized using spaCy tokenizer, special characters were removed as well as newline characters, multiple spaces were compressed to one space and capital letters were replaced by a lowercase equivalent. Finally, tokens were represented using FastText embeddings (Bojanowski et al., 2016) with the size of 300. The data was

<sup>1</sup><https://github.com/akashkm99/Interpretable-Attention> at commit 2d8dd37.

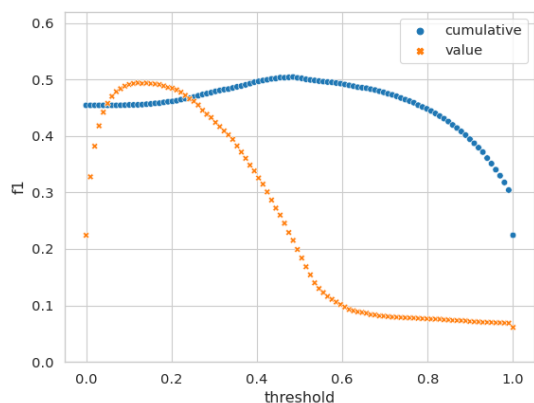


Figure 1: The chart of span-level F1 score with respect to the threshold for OrthoLSTM. Results obtained on the validation set.

split into training, validation and testing sets with the 80:10:10 ratio.

The model consisted of 1-layered orthogonal LSTM followed by the dense layer and was trained using Adam optimizer with learning rate of  $1e-3$ , weight decay of  $1e-5$  and the batch size 32. This model has obtained 0.957 ROC AUC for comment-level classification. The performance of the model proved it to be sufficient for further analysis of the attention. The key parameter to obtain token-level prediction was setting a threshold for toxic token selection based on the attention value. Two approaches were investigated:

- value-based: all tokens with an attention score higher than the certain threshold were selected as toxic
- cumulative: sorting the tokens by the highest attention score, each token was added to the toxic set as long as the cumulative value of attention was not covered (e.g. 70% of the whole model’s attention)

The span-level F1 score on the validation set with different threshold values for both approaches is presented in Figure 1. The best results were obtained by 0.12 threshold for the value method and 0.48 threshold for the cumulative method.

### 3.2 SHAP

Shapley Additive Explanations (Lundberg and Lee, 2017) method has been introduced as a model-agnostic framework that provides interpretation for model predictions. The authors of SHAP recognise

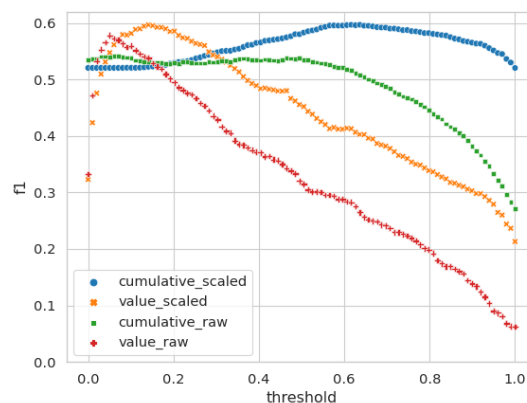


Figure 2: The chart of span-level F1 score with respect to the threshold for SHAP. Results obtained on the validation set.

that common feature importance measures rely on the same explanation model and propose a new method consisting of Shapley values approximated with kernel methods e.g. LinearLIME.

The authors decided to apply SHAP framework to explain the BERT model, trained for toxic comment classification. An implementation provided by Hanu and Unitary team (2020) in Detoxify<sup>2</sup> repository was used. The model is the `bert-base-uncased`<sup>3</sup> from `transformers` library trained on Wikipedia Comments (Wulczyn et al., 2017) using Adam optimizer with learning rate  $3e-5$  and weight decay  $3e-6$ . The model obtains 0.909 ROC AUC in the binary classification task on CivilComments dataset (Borkan et al., 2019).

As in the previous method, the problem of threshold selection has been a significant bottleneck of this approach as well. The difference, as opposed to LSTM with attention, is that SHAP scores do not sum up to 1. This allowed to treat the values in two more ways: operating on unchanged values or rescaling them to sum up to 1 and performing operations in the previous fashion. The results are presented in Figure 2. Overall, the scaled approaches outperformed methods using raw SHAP scores.

### 3.3 Token classification

As opposed to explanation-based solutions presented in the previous section, the authors also

<sup>2</sup><https://github.com/unitaryai/detoxify> at commit 18fd29e.

<sup>3</sup><https://huggingface.co/bert-base-uncased>

tested a classic, supervised learning approach to the task. This solution is based on pre-trained BERT model (Devlin et al., 2018), which is fine-tuned to predict toxic spans.

The trial part of the dataset was used for validation, where the training and testing parts were used according to their purpose. Due to the size of the training set, a model as complex as BERT is prone to overfitting. Therefore, the training time of the model consisted of 3 epochs and the learning rate started from  $4.7e-5$  and was divided by 10 after each epoch. The authors also proposed data augmentation to deal with this problem which was described in details in Section 3.3.3. As optimizer AdamW was used and batch size was set to 8.

### 3.3.1 Dealing with long inputs

The dataset for the task contained many comments with a noticeably large number of tokens. However, the BERT model is limited to 512 tokens for input data. The problem was solved by reformatting the input to fit the given size. A sample was built by adding whole sentences from the input as long as the size limit was not exceeded. A set of possible breakpoints consisted of end of sentence positions to ensure that each sentence is not split in half. Chosen breakpoints and sentence IDs were remembered in order to calculate the span-level F1 score. The length of the preprocessed samples did not have to be based only on BERT limitations and can be adjusted for better training complexity. The shorter splits were tested as a way to speed up the learning process. The results were presented in Table 1. The training time was reduced by 40%, but the decrease in quality of prediction appeared due to the much narrower context fed into the model.

	Time [m]		span-level F1
Tokens	128	43	0.660
	512	72	0.672

Table 1: The learning time in minutes and span-level F1 score on validation set according to the length of input in tokens.

### 3.3.2 BERT extensions comparison

Furthermore, given promising results obtained by BERT, the authors have decided to compare it to other BERT-based models. The models selected for testing were ELECTRA (Clark et al., 2020), XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019), SqueezeBERT (Iandola et al., 2020) and

for each of them the implementation from the `transformers`<sup>4</sup> library was taken. The models were tested in the same manner as BERT. The hyperparameters learning rate, batch size etc. were also the same. The results are presented in Table 2. One can notice that models with a higher number of parameters tend to work better. However, learning curves analysis indicates that they are also more prone to overfitting.

Model	span-level F1
XLNet	0.678
RoBERTa	0.676
BERT	0.672
SqueezeBERT	0.657
ELECTRA	0.646

Table 2: The comparison of BERT and BERT-based models on the validation dataset.

### 3.3.3 Data augmentation

In order to deal with overfitting, the authors decided to apply a simple augmentation method. The augmentation is performed by random swaps of words appearing in text no further than 3 words from each other. The number of swaps depends on the length of the input, for each input it is calculated as follows:  $alpha * number\_of\_tokens$ . The results obtained while using augmentation presented in Table 3 are ambiguous and do not give a clear answer whether it helps or not. However, after a thorough analysis of train vs. loss curves, the authors noticed that the model does not overfit quite as much and the metric have smaller fluctuations. Therefore, the authors decided to use this technique during further training whenever it helps on the validation dataset.

alpha	span-level F1
0	0.672
0.2	0.669
0.5	0.675

Table 3: The span-level F1 score on validation set for BERT model with augmentation with a given alpha.

### 3.3.4 Filling empty spaces in between spans

While analysing the labels provided for the training set, the authors noticed that at times, whole text

<sup>4</sup><https://huggingface.co/transformers> at version 4.0.1

fragments were labelled as toxic, including spaces in between words. Further analysis showed that 9.54% spans in the training set are spaces. Due to the fact that tokenization resulted in omitting those spaces, an experiment had been performed in order to deal with this problem. If the output contained toxic spans separated by one or two characters, the character was also considered toxic. This technique slightly improved the performance as noted in Table 4, therefore the authors decided to include it in the final solution.

Chars	span-level F1
0	0.6720
1	0.6735
2	0.6730

Table 4: The results of marking the given number of characters in between toxic spans as toxic by BERT on validation set.

### 3.3.5 Ensemble

Models previously described in Section 3.3.2 were concatenated into an ensemble. The aggregation of predictions was performed with hard voting - a span was considered toxic only when 3 out of 5 models returned a toxic prediction.

## 4 Results

The results obtained by different methods on the test set are presented in Table 5. The highest performing model in this paper turned out to be an ensemble of different BERT-based models scoring **13th out of 91** solutions. The source code for all approaches is publicly available on GitHub<sup>5</sup>.

<sup>5</sup><https://github.com/hancia/ToxicSpansDetection>

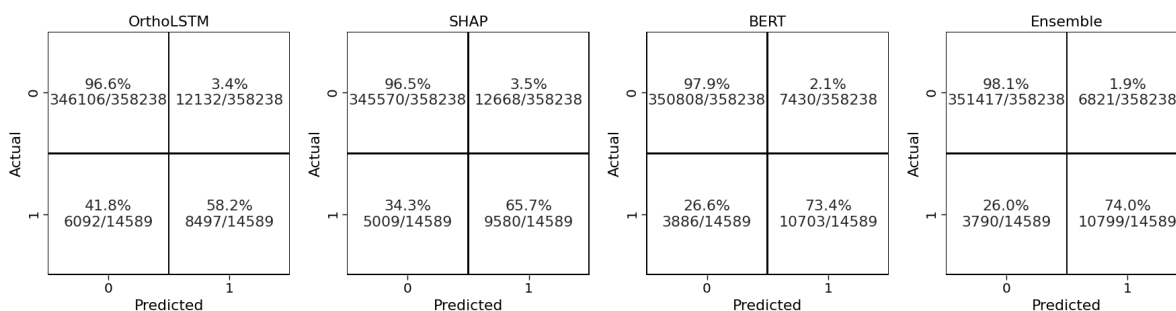


Figure 3: Confusion matrices for each method. 1 refers to toxic class, non-toxic samples are marked by 0.

Method	span-level f1
OrthoLSTM	0.4970
SHAP	<b>0.5987</b>
BERT	0.6513
XLNet	0.6624
BERT + aug 0.5 + fill 1	0.6780
Ensemble	<b>0.6859</b>

Table 5: The comparison of selected models' performance on the test dataset. Ensemble consists of all models stated in the section 3.3.2 - BERT was used with augmentation ( $\alpha=0.5$ ) and filling chars ( $char=1$ ), other models were only used with filling chars ( $char=1$ ).

## 5 Discussion

As presented in the previous section, the supervised solution to token classification outperformed both xAI approaches. The BERT model was trained specifically to solve this type of problem and therefore achieved a score slightly better than LSTM or SHAP. High labelling costs need to be taken into account when comparing those solutions, as training a robust BERT model would require much more data, that would not be necessary for xAI approaches, which are generally unsupervised.

Surprisingly, a model-agnostic approach turned out to perform much better than the attention-based solution, even though the explained models achieved very similar results on the toxic comment detection task. In Figure 3 one can see that the number of false negatives was significantly higher while relying on attention than it was while using Shapley values. This could be because the model might pay a significant portion of attention to a very toxic word, which is enough to recognise the comment as toxic. While analysing the model's focus can improve the understanding of how the model works, it might not necessarily be enough to

translate into clear decisions for each of the input components. Furthermore, the need for threshold tuning somehow forced explainable approaches to mark a certain number of tokens as toxic, which could be reflected in a slightly higher number of false positives. Visualisations of example predictions can be seen in Table 6.

While xAI approaches might not be fitted to solve the problem of predicting the toxicity of each input word, they might still be useful for improving the transparency and understanding of predictions made by comment-level models. As recognised in error analysis, the number of false negatives was significantly higher for LSTM and SHAP. But in terms of explanation, it might be enough for the user to obtain the few most toxic words per comment, rather than marking all of them, no matter how low the toxicity score is. This would not only provide the explanation on what the model considered while making a prediction but would also be a clear and transparent answer for the user in many real-time use cases. Further examination might need to be done in order to assess the performance of those methods for a task specified in the aforementioned way.

## 6 Conclusions

This work discussed different approaches to the Toxic Spans Detection task. Supervised toxic token classification and xAI methods were examined to compare the results and assess whether explaining high-performing models can lead to a similar quality of prediction as models dedicated to a more detailed task. The supervised approach using the BERT model achieved the best result in this task, but the xAI methods have proven to be an interesting alternative that could reduce data preparation costs and improve transparency and understanding of the model's predictions. While not currently outperforming BERT, explainable methods can be sufficient for many tasks where binary decision models are used. A system for toxic spans detection was prepared, achieving a 0.6859 span-level F1 score and placing 13th out of 91 in the overall ranking.

## Acknowledgments

The authors are grateful to Mateusz Lango for his invaluable guidance. The research by Kamil Pluciński was partially supported by TAILOR, a project funded by EU Horizon 2020 research and

innovation programme under GA No 952215.

## References

- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. 2016. [Enriching word vectors with subword information](#). *CoRR*, abs/1607.04606.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Electra: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- William G. Cochran. 1977. *Sampling Techniques, 3rd Edition*. John Wiley.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Reza Ghaeini, Xiaoli Fern, and Prasad Tadepalli. 2018. [Interpreting recurrent and attention-based neural models: a case study on natural language inference](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4952–4957, Brussels, Belgium. Association for Computational Linguistics.
- Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.
- Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, and Kurt W. Keutzer. 2020. [Squeezebert: What can computer vision teach nlp about efficient neural networks?](#) In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 124–135. Association for Computational Linguistics.
- Sarthak Jain and Byron C. Wallace. 2019. [Attention is not explanation](#). *CoRR*, abs/1902.10186.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).
- Scott Lundberg and Su-In Lee. 2017. [A unified approach to interpreting model predictions](#). *CoRR*, abs/1705.07874.
- Akash Kumar Mohankumar, Preksha Nema, Sharan Narasimhan, Mitesh M. Khapra, Balaji Vasan Srinivasan, and Balaraman Ravindran. 2020. [Towards transparent and explainable attention models](#). In

*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4206–4216, Online. Association for Computational Linguistics.

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.

Felix Richter. 2020. [Toxic content runs rampant on facebook](#).

We Are Social. 2020. [Digital in 2020: Global digital overview](#).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sarah Wiegrefe and Yuval Pinter. 2019. [Attention is not not explanation](#).

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#). In *Proceedings of the 26th International Conference on World Wide Web, WWW '17*, pages 1391–1399, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

## A Examples

OrthoLSTM	SHAP	Ensemble
That's right. They are not normal. And I am starting from the premise that they are <b>ABNORMAL</b> . Proceed with the typical <b>racist</b> , <b>bigot</b> , <b>sexist rubbish</b> . Thanks!	That's right. They are not normal. And I am starting from the premise that they are <b>ABNORMAL</b> . Proceed with the typical racist, bigot, <b>sex ist</b> , <b>rubbish</b> . Thanks!	That's right. They are not normal. And I am starting from the premise that they are <b>ABNORMAL</b> . Proceed with the typical <b>racist</b> , <b>bigot</b> , <b>sexist rubbish</b> . Thanks!
A rock has more <b>mental</b> horsepower than the indubitably <b>ignorant</b> . And more integrity as well. They're largely very <b>dishonest people</b> who repeat the same <b>lies</b> over and over. Just like their hero.	A rock has more mental horsepower than the indubitably <b>ignorant</b> . And more integrity as well. They're largely very <b>dishonest people</b> who repeat the same lies over and over. Just like their hero.	A rock has more mental horsepower than the indubitably <b>ignorant</b> . And more integrity as well. They're largely very <b>dishonest people</b> who repeat the same lies over and over. Just like their hero.
Ha ha, HILLARY LOST. All you hillary <b>fools</b> need to be rounded up and put in prison along with your leader. You are all anti-American scum. Perhaps we should <b>dump</b> you in Mexico along with all the illegal <b>trash</b> you let in...if you love 'em so much, go live with 'em. But you are not welcome here, you hate filled Marxist <b>morons</b> .	Ha ha, HILLARY LOST. All you hillary <b>fools</b> need to be rounded up and put in prison along with your leader. You are all anti-American <b>scum</b> . Perhaps we should <b>dump</b> you in Mexico along with all the illegal <b>trash</b> you let in...if you love 'em so much, go live with 'em. But you are not welcome here, you hate filled Marxist <b>morons</b> .	Ha ha, HILLARY LOST. All you hillary <b>fools</b> need to be rounded up and put in prison along with your leader. You are all anti-American <b>scum</b> . Perhaps we should <b>dump</b> you in Mexico along with all the illegal <b>trash</b> you let in...if you love 'em so much, go live with 'em. But you are not welcome here, you hate filled Marxist <b>morons</b> .
Rabidly <b>anti-Canadian</b> troll.	Rabidly <b>anti -Canadian</b> troll.	Rabidly <b>anti-Canadian</b> troll.
No, my poor <b>benighted</b> <b>correspondent</b> , your reductionism reveals an archaic mindset. Intersex, questioning, nongendered, and a vast variety of others have rights too!	No, my <b>poor</b> <b>benighted</b> correspondent, your reductionism reveals an archaic mindset. Intersex, questioning, nongendered, and a vast variety of others have rights too!	No, my poor <b>benighted</b> correspondent, your reductionism reveals an archaic mindset. Intersex, questioning, nongendered, and a vast variety of others have rights too!
justin is an <b>embarrassment</b> to Canada. he needs a muzzle. he needs a <b>brain</b> .	justin is an <b>embarrassment</b> to Canada. he needs a muzzle. he needs a <b>brain</b> .	justin is an <b>embarrassment</b> to Canada. he needs a muzzle. he needs a brain.
You are a <b>simpleton</b> , who is full of racial hatred. That is too bad. You make America a worse place.	You are a <b>simpleton</b> , who is full of <b>racial</b> <b>hatred</b> . That is too bad. You make America a worse place.	You are a <b>simple</b> <b>ton</b> , who is full of racial hatred. That is too bad. You make America a worse place.
When you consider the source - he writes like the Trump we've all come to know - "I could stand in the middle of 5th Avenue and <b>shoot</b> somebody and I wouldn't lose voters", a <b>racist</b> , <b>misgynistic</b> , <b>liar</b> who only brings hate to the table.	When you consider the source - he writes like the Trump we've all come to know - "I could stand in the middle of 5th Avenue and shoot somebody and I wouldn't lose voters", a <b>racist</b> , <b>misgynistic</b> , <b>liar</b> who only brings hate to the table.	When you consider the source - he writes like the Trump we've all come to know - "I could stand in the middle of 5th Avenue and shoot somebody and I wouldn't lose voters", a <b>racist</b> , <b>mis gy nistic</b> , <b>liar</b> who only brings hate to the table.
<b>Total</b> <b>rubbish</b> ! The birther bit was started by Crooked Hillary and perpetuated by Zero his own <b>damn</b> self.	<b>Total</b> <b>rubbish</b> ! The birther bit was started by Crooked Hillary and perpetuated by Zero his own <b>damn</b> self.	<b>Total</b> <b>rubbish</b> ! The birther bit was started by Crooked Hillary and perpetuated by Zero his own <b>damn</b> self.
<b>Damn</b> , you <b>beat</b> me to it	<b>Damn</b> , you beat me to it	<b>Damn</b> , you beat me to it
I don't think they eat them, just <b>kill</b> <b>them</b> , <b>chop</b> them up and sell off the parts.	I don't think they eat them, just <b>kill</b> <b>them</b> , <b>chop</b> them up and sell off the parts.	I don't think they eat them, just <b>kill</b> <b>them</b> , <b>chop</b> them up and sell off the parts.
F*cking <b>nasty</b> ...	F*ck <b>ing</b> nasty...	F*cking <b>nasty</b> ...

Table 6: Example predictions done by models. No background colour refers to true negative, green - true positive, red - false negative and blue - false positive. Examples were selected where the predictions between models were different.

# GoldenWind at SemEval-2021 Task 5: Orthrus – An Ensemble Approach to Identify Toxicity

Marco Palomino\*

Dawid Grad†

James Bedwell‡

School of Engineering, Computing and Mathematics

University of Plymouth

Plymouth, Devon, PL4 8AA, United Kingdom

marco.palomino@plymouth.ac.uk\*

{dawid.grad†, james.bedwell‡}@students.plymouth.ac.uk

## Abstract

Many new developments to detect and mitigate toxicity are currently being evaluated. We are particularly interested in the correlation between toxicity and the emotions expressed in online posts. While toxicity may be disguised by amending the wording of posts, emotions will not. Therefore, we describe here an ensemble method to identify toxicity and classify the emotions expressed on a corpus of annotated posts published by Task 5 of SemEval 2021—our analysis shows that the majority of such posts express anger, sadness and fear. Our method to identify toxicity combines a lexicon-based approach, which on its own achieves an F1 score of 61.07%, with a supervised learning approach, which on its own achieves an F1 score of 60%. When both methods are combined, the ensemble achieves an F1 score of 66.37%.

## 1 Introduction

Healthy conversations are only possible when people feel safe from abuse and do not resort to using violent language. Regrettably, violent and inflammatory language is becoming increasingly common online. Indeed, the rhetoric of violence recently employed on social media has persuaded platforms, such as Twitter, to create new policies to prevent the use of threatening language (Twitter, Inc., 2021). A jargon word, *cyberbullying*, has been coined lately to refer to the use of electronic communication to send or post messages of an intimidating or threatening nature (Zaheri et al., 2020).

Along with cyberbullying, other forms of verbal abuse employed on social media, such as online harassment and hate speech, are now being collectively referred to as *toxicity* in language (Mohan et al., 2017). We are interested in developing algorithms to recognise toxicity and measure its impact on the sentiment expressed.

Most of the data available to investigate toxicity classify whole comments or documents (Wulczyn et al., 2017; Borkan et al., 2019), and do not identify “spans”—that is, the precise word sequences that make a text toxic. Given how important such spans are for the implementation of semi-automated moderation, we have participated on Task 5 (Toxic Spans Detection) of the *International Workshop on Semantic Evaluation (SemEval) 2021* (Pavlopoulos et al., 2021). Thus far, we have concentrated on the combination of two approaches: a lexicon-based approach and a supervised learning approach to identify toxic spans.

Although the identification of toxic spans in online posts can be aided by a suitable lexicon of toxic words, such words can easily be concealed through minor changes—for instance, “fck urself” is a toxic span that would evade detection based on basic lists of profane words. However, emotions are harder to conceal. Hence, we are interested in using opinion mining to uncover the emotions expressed in text. Emotions may be able to identify toxicity, regardless of wordings and spellings. Thus, we dedicate part of this study to measure the correlation between toxicity and emotions.

The remainder of this paper is organised as follows: Section 2 reviews the related work. Section 3 describes the datasets that we used for our experimentation. Section 4 is dedicated to explain our algorithm for the identification of toxic spans. Section 5 presents our results and, finally, Section 6 offers our conclusions.

## 2 Background

The existing literature on toxicity focuses on two main aspects: the compilation and annotation of corpora for research purposes (Fortuna et al., 2020; Waseem, 2016); and the automatic detection of different types of toxic text.



Among the different types of toxic text under scrutiny, we may include hate speech (Badjatiya et al., 2017; Davidson et al., 2017; Del Vigna et al., 2017), online harassment (Golbeck et al., 2017), racism (Waseem, 2016), sexism (Jha and Mamidi, 2017), abusive language (Mehdad and Tetreault, 2016) and cyberbullying (Zhong et al., 2016).

At present, the detection of toxicity is largely based on state-of-the-art natural language processing techniques, typically involving machine learning. The main drawback of such techniques resides in the limited generalisation potential of trained machine learning models (Fortuna et al., 2020). To overcome this weakness, we have integrated into our research the use of a lexicon-based approach, where toxic language is identified with the help of a dictionary of words associated with toxic text (De Smedt et al., 2020).

Combining lexicons with machine learning approaches has already been evaluated by other researchers, remarkably Pamungkas and Patti (Pamungkas and Patti, 2019), though they employed a lexicon originally built for the Italian language, and then translated it into other languages, whereas we focus on English from the start. Various other lexicons, handcrafted by domain experts who specialise on the identification of toxicity have been published too—for example, *Textgain’s Profanity and Offensive Words* lexicon (De Smedt et al., 2020)—but many of them are not available for free.

In an attempt to mitigate toxicity and promote work on this area, the research community has released a number of annotated datasets for investigating different forms of toxicity (Waseem and Hovy, 2016; Waseem, 2016; Golbeck et al., 2017). However, they all follow different labelling conventions. Consequently, they cannot be analysed using a uniform method.

Overall, toxicity detection and classification lacks a consistently labelled standard dataset for comparative evaluation (Schmidt and Wiegand, 2017). Therefore, the data provided by Task 5 (Toxic Spans Detection) of SemEval 2021 is very well regarded (Pavlopoulos et al., 2021).

### 3 Experimental Setup

Task 5 of SemEval 2021 uses posts from the publicly available *Civil Comments* dataset (TensorFlow, 2021). Such a dataset comprises annotations indicating which entire posts are toxic, but it does not label particular toxic spans within the posts.

The Civil Comments platform (Drupal, 2021), which is where the posts come from, is a commenting plugin for independent news websites. All the comments were created between 2015 and 2017, and they appeared on approximately 50 English language websites across the world. When Civil Comments shut down in 2017, the comments became publicly available in an open archive for future research (TensorFlow, 2021).

To build the dataset, SemEval retained only posts that were found toxic—or severely toxic—by at least half of the annotators involved in Borkan, *et al.*’s annotation (Borkan et al., 2019). This comprises 30k toxic posts, approximately, out of the original 1.2M. Then, a random subset of 10k posts from these 30k toxic posts were chosen for toxic spans annotation (CodaLab, 2021).

## 4 System Overview

Although machine learning technology is being widely employed to detect toxic text automatically, the use of a lexicon to identify and prevent toxicity in social media still constitutes a valuable approach. Indeed, the number of lexicons specialised on the detection of profanity, offensive speech and toxicity in general has grown steadily in recent times (De Smedt et al., 2020).

Lexicons are not susceptible to algorithmic bias (Hajian et al., 2016), and are not limited to the domain and scope of the training data, which has previously raised a number of ethical concerns, given how much training data is historically associated with particular communities (Hao, 2019). Hence, we employ a lexicon as our first step in the detection of toxic spans.

Originally, our lexicon was made, specifically, for Task 5 of SemEval 2021, as we compiled it by extracting all the toxic words available in the training and trial datasets for Task 5—we considered a word as a *toxic* word if it was included in a toxic span identified by the annotators.

Upon compiling all the toxic words available in the training and trial datasets (1,287 words), we proceeded to extend our lexicon with words listed in other lexicons. While there are many freely-available lexicons of toxic words, we favoured those that maintained the accuracy of the detection of toxic spans achieved by our lexicon. Table 1 shows the F1 scores achieved by each of the lexicons considered, when combined with our lexicon to evaluate them on the training dataset.

Lexicon	F1 Training	Number of words
Task 5 Lexicon	64.30%	1,287
<b>Banned Word List</b>	<b>64.30%</b>	1,332
Offensive/Profane Word List	61.25%	2,516
<b>Google’s Profanity Words</b>	<b>64.30%</b>	1,681
Insult.wiki	63.94%	1,846
Compiled_bad_words	63.99%	2,546
<b>Swear Word List &amp; Curse Filter</b>	<b>64.30%</b>	1,580

Table 1: F1 score per lexicon (evaluated on the training dataset).

The first row of Table 1 refers to the lexicon we created after compiling all the toxic words available in the training and trial datasets of Task 5 of SemEval 2021—we named this lexicon the *Task 5 Lexicon*. Using bold font, we have highlighted the details of the lexicons that achieved the same F1 score as the Task 5 Lexicon, when combined with it to evaluate them on the training dataset. Such lexicons are the ones that we decided to use, namely, the *Banned Word List* (<http://www.bannedwordlist.com/>), *Google’s Profanity Words* (<https://github.com/RobertJGabriel/Google-profanity-words>), and the *Swear Word List & Curse Filter* (<https://www.noswearing.com/dictionary>). Table 1 also displays the number of words available in each of the lexicons evaluated, when combined with the Task 5 Lexicon.

As shown in Table 1, the *Offensive / Profane Word List* (<https://www.cs.cmu.edu/~biglou/resources/>) and the *Compiled\_bad\_words* ([https://github.com/minerva-ml/open-solution-toxic-comments/blob/master/external\\_data/compiled\\_bad\\_words.txt](https://github.com/minerva-ml/open-solution-toxic-comments/blob/master/external_data/compiled_bad_words.txt)) have a negative impact on the performance of toxicity detection, even if it is only by a small margin. Thus, we discarded these lexicons.

After creating our lexicon, we manually removed from it words that were part of the toxic spans annotated in Task 5 of SemEval 2021, but were not included in the three lexicons displayed in bold font in Table 1. For example, the word “mistake” located in the post “They elected Trump, which was certainly a mistake” was considered toxic by the annotators, in the context of the post. However, we removed it from our lexicon, because “mistake” does not appear in any of the three lexicons mentioned above. Our lexicon comprises a total of 1,929 words, and we refer to it as the *Orthrus* lexicon—it is available at <https://github.com/Orthrus-Lexicon/Toxic>.

While our lexicon-based approach was considerably useful to identify toxicity, as we will show in Section 5, we recognise the value of machine learning approaches. The success of the *Perspective* project undertaken by Google and Jigsaw to rate toxicity by means of machine learning (Jain et al., 2018), as well as the impact of the *Perspective API* to mitigate toxicity using machine learning certainly deserve our attention. Therefore, we opted to employ *spaCy* (Explosion, 2021b), an open-source software library for natural language processing, to develop a supervised learning approach for the identification of toxicity.

Our choice of *spaCy* was further motivated by the organisers of Task 5 of SemEval 2021, who released a Python script referring, precisely, to this library (Task 5, 2021). Initially, we employed `en_core_web_sm`, which is a *spaCy* model for the English language (Explosion, 2021a). We employed this model, because it was the one used in the code provided by the organisers of Task 5 as a solution for some NLP tasks—namely, POS tagging, NER and dependency parsing (Task 5, 2021). However, given that `en_core_web_sm` is based on a small English corpus, we also tested `en_core_web_lg`, which is *spaCy*’s large English model (Explosion, 2021a).

Despite *spaCy*’s large English model being understandably slower, it did not appear to improve the performance of our implementation. The F1 score achieved, on average, by *spaCy*’s small English model after 10 executions (59.61%) was approximately the same as the score achieved by the large English model under the same circumstances (59.95%). Thus, we favoured the choice of the small model, as it was faster to train.

## 5 Results

Table 2 shows the F1 score achieved by our implementation when evaluating it on the test dataset.

Approach	F1 Score
Orthrus Lexicon	61.07%
Orthrus Lexicon + spaCy Model (Union)	61.53%
Orthrus Lexicon + spaCy Model (Intersection)	66.37%

Table 2: Evaluation on the test dataset.

As shown in Table 2, our lexicon achieves, on its own, an F1 score of 61.07%. By combining our lexicon with the supervised learning approach implemented using spaCy, we achieve two results: 61.53%, if we consider the union of the results yielded by the lexicon and the supervised learning approach; and 66.37%, if we consider the intersection of the results yielded by the lexicon and the supervised learning approach.

We are interested in the identification of emotions expressed in text, because concealing emotions may be harder than disguising toxicity. For example, the post “*uh, no, he’s a belligerent buffoon (and a traitor)*”, which is post 1,928 of the training dataset of Task 5 of SemEval 2021, lacks any recognisable toxic features, such as insults or swear words. Hence, it is classified as non-toxic by any of the lexicons highlighted in bold font in Table 1. Moreover, this post does not have any toxic spans marked by the annotators. Nevertheless, the negative sentiment of “belligerent buffoon” and “traitor”—words which are not typically found in any abusive word list—guarantees that the message conveyed is definitely toxic; otherwise, it would not be part of the training dataset.

If we included emotion information in our analysis, we could immediately detect the negative tone of the post mentioned above. Indeed, the probability of such a post to communicate anger is 1.0, according to `text2emotion`, a Python package to extract emotions from text (Python Software Foundation, 2021). The expression of anger is so evident in this case that the post can be marked as a candidate to be considered toxic.

Using `text2emotion`, we assigned each post in the test dataset a probability associated with each of the emotions reported in Figure 1. The values shown in Figure 1 represent the addition of the probabilities of each emotion to occur in each of the posts of the test dataset. Clearly, fear, sadness and anger—the three emotions combined together—are more likely to occur than happiness and surprise—the two emotions combined together—which may characterise the toxicity of the dataset.

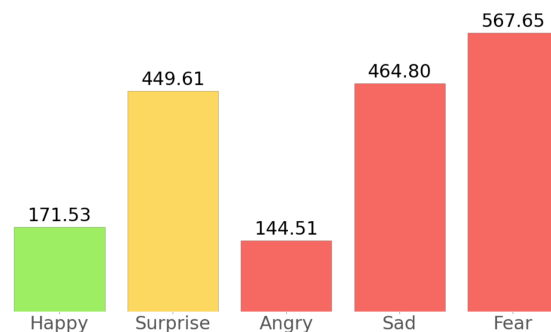


Figure 1: Emotion expressed on the test dataset.

## 6 Conclusions

In this paper, we have described the creation of a lexicon of toxic words and a supervised learning approach to identify toxicity in online posts. Our lexicon, along with the supervised learning approach, achieved an F1 score of 66.37% on Task 5 of SemEval 2021. We have also explored the relationship between emotions and toxicity. Although our study is still in progress, preliminary results indicate that there exists a correlation between emotions such as sadness and fear and toxicity.

## References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web*, pages 759–760.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification. In *Proceedings of the 2019 World Wide Web Conference*, pages 491–500.
- CodaLab. 2021. SemEval 2021 Task 5: Toxic Spans Detection. [https://competitions.codalab.org/competitions/25623#learn\\_the\\_details-data](https://competitions.codalab.org/competitions/25623#learn_the_details-data).
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the International AAAI Conference on Web and Social Media*.

- Tom De Smedt, Pierre Voué, Sylvia Jaki, Melina Röttcher, and Guy De Pauw. 2020. Profanity & Offensive Words (POW). *Textgain*.
- Fabio Del Vigna, Andrea Cimino, Felice Dell’Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate Me, Hate Me Not: Hate Speech Detection on Facebook. In *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, pages 86–95.
- Drupal. 2021. Civil Comments. <https://www.drupal.org/project/civilcomments>.
- Explosion. 2021a. English – Available Trained Pipelines for English. <https://spacy.io/models/en>.
- Explosion. 2021b. spaCy. <https://spacy.io/>.
- Paula Fortuna, Juan Soler, and Leo Wanner. 2020. Toxic, Hateful, Offensive or Abusive? What Are We Really Classifying? An Empirical Analysis of Hate Speech Datasets. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6786–6794.
- Jennifer Golbeck, Zahra Ashktorab, Rashad O Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A Geller, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, et al. 2017. A Large Labeled Corpus for Online Harassment Research. In *Proceedings of the 2017 ACM on Web Science Conference*, pages 229–233.
- Sara Hajian, Francesco Bonchi, and Carlos Castillo. 2016. Algorithmic Bias: From Discrimination Discovery to Fairness-Aware Data Mining. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2125–2126.
- Karen Hao. 2019. AI Is Sending People to Jail—And Getting It Wrong. *MIT Technology Review*.
- Edwin Jain, Stephan Brown, Jeffery Chen, Erin Neaton, Mohammad Baidas, Ziqian Dong, Huanying Gu, and Nabi Sertac Artan. 2018. Adversarial Text Generation for Google’s Perspective API. In *International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1136–1141. IEEE.
- Akshita Jha and Radhika Mamidi. 2017. When Does a Compliment Become Sexist? Analysis and Classification of Ambivalent Sexism Using Twitter Data. In *Proceedings of the Workshop on NLP and Computational Social Science*, pages 7–16.
- Yashar Mehdad and Joel Tetreault. 2016. Do Characters Abuse More than Words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303.
- Shruthi Mohan, Apala Guha, Michael Harris, Fred Popowich, Ashley Schuster, and Chris Priebe. 2017. The Impact of Toxic Language on the Health of Reddit Communities. In *Canadian Conference on Artificial Intelligence*, pages 51–56. Springer.
- Endang Wahyu Pamungkas and Viviana Patti. 2019. Cross-Domain and Cross-Lingual Abusive Language Detection: A Hybrid Approach with Deep Learning and a Multilingual Lexicon. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 363–370.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 Task 5: Toxic Spans Detection (To Appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Python Software Foundation. 2021. text2emotion 0.0.5. <https://pypi.org/project/text2emotion/>.
- Anna Schmidt and Michael Wiegand. 2017. A Survey on Hate Speech Detection Using Natural Language Processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.
- Task 5. 2021. Toxic Spans 2021. <https://groups.google.com/g/toxic-spans>.
- TensorFlow. 2021. CivilComments Dataset. [https://www.tensorflow.org/datasets/catalog/civil\\_comments](https://www.tensorflow.org/datasets/catalog/civil_comments).
- Twitter, Inc. 2021. Violent Threats Policy. <https://help.twitter.com/en/rules-and-policies/violent-threats-glorification>.
- Zeerak Waseem. 2016. Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter. In *Proceedings of the First workshop on NLP and Computational Social Science*, pages 138–142.
- Zeerak Waseem and Dirk Hovy. 2016. Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex Machina: Personal Attacks Seen at Scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.
- Sara Zaheri, Jeff Leath, and David Stroud. 2020. Toxic comment classification. *SMU Data Science Review*, 3(1):13.
- Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 3952–3958.

# LISAC FSDM USMBA at SemEval-2021 Task 5: Tackling Toxic Spans Detection Challenge with Supervised SpanBERT-based Model and Unsupervised LIME-based Model

Abdessamad Benlahbib<sup>1\*</sup>, Ahmed Alami<sup>2</sup>, Hamza Alami<sup>2\*</sup>,

<sup>1</sup> LISAC Laboratory, Faculty of Sciences Dhar EL Mehraz (F.S.D.M),  
Sidi Mohamed Ben Abdellah University (U.S.M.B.A)

<sup>2</sup> Laboratory of Engineering Sciences, National School of Applied Sciences,  
Ibn Tofail University, Kenitra, Morocco

abdessamad.benlahbib@usmba.ac.ma, alami.alami1996@gmail.com,  
hamza.alami1@usmba.ac.ma

## Abstract

Toxic spans detection is an emerging challenge that aims to find toxic spans within a toxic text. In this paper, we describe our solutions to tackle toxic spans detection. The first solution, which follows a supervised approach, is based on SpanBERT model. This latter is intended to better embed and predict spans of text. The second solution, which adopts an unsupervised approach, combines linear support vector machine with the Local Interpretable Model-Agnostic Explanations (LIME). This last is used to interpret predictions of learning-based models. Our supervised model outperformed the unsupervised model and achieved the f-score of 67,84% (ranked 22/85) in Task 5 at SemEval-2021: Toxic Spans Detection.

## 1 Introduction

By dint of the massive production of user-generated content in social media, moderation becomes crucial to promote healthy online discussions by removing toxic posts and contents. However, it is nearly impossible for a human to keep tracking user-generated content. Thus, the need for the right tools and technologies to help in such a task becomes a necessity.

The Toxic Spans Detection task aims to detect the spans that make a text toxic. While several toxicity detection datasets (Wulczyn et al., 2017; Borkan et al., 2019) and models (Pavlopoulos et al., 2017a, 2019; Schmidt and Wiegand, 2017; Pavlopoulos et al., 2017b; Zampieri et al., 2019; Alami et al., 2020) have been released. However, these works estimate the likelihood of a document being toxic with weak interpretability. In fact, highlighting toxic spans can assist human moderators who often deal with lengthy comments, and who prefer attribution instead of just a system-generated unexplained toxicity score per post.

\*contributed equally

In this paper, we propose two solutions to tackle toxic spans detection (Pavlopoulos et al., 2021). The first solution, which follows a supervised approach, is based on SpanBERT (Joshi et al., 2020) model that is pre-trained on span boundary objective and considers masks contiguous spans. Therefore, SpanBERT gives better spans representations and predictions. The second solution, which adopts an unsupervised approach, combines linear support vector machine (Fan et al., 2008) with the Local Interpretable Model-Agnostic Explanations (LIME) (Ribeiro et al., 2016). LIME is an explanation technique that seeks to faithfully interpret the predictions of any classifier.

This paper is organized as follows: Section 2 describes the proposed methods; Section 3 presents the experimental results; Finally, Section 4 concludes and outlines future directions.

## 2 Methods

In this section, we describe the proposed solutions including SpanBERT-based method which is based on supervised approach, and SVM and LIME-based method that is based on unsupervised approach.

### 2.1 SpanBERT-based method

We use SpanBERT (Joshi et al., 2020) a pre-trained model built to improve spans of text representation and prediction. It differs from BERT (Devlin et al., 2019) as it (1) masks contiguous random spans, instead of random tokens; and (2) is trained on span-boundary objective, i.e., the model is optimized to predict the masked span given tokens at its boundary. We considered the toxic span text detection as a sequence labeling task. Thus, we performed a transformation to the dataset and fine-tuned SpanBERT to this specific task.

### 2.1.1 Data preparation

The raw dataset consists of a set of toxic texts where each element is annotated with an array that contains characters' indices. These indices are considered as the toxic span of text. In order to train SpanBERT on this dataset, we applied the pre-trained SpanBERT tokenizer to tokenize sentences, and we built the target arrays by annotating words that contain toxic characters' indices. For instance, given a sentence that contains  $n$  tokens, then the target array contains  $n$  elements, where the elements that contain a toxic character are labeled as positive "1" otherwise they are labeled as negative "0". Figure 1 illustrates the pipeline of dataset preparation.

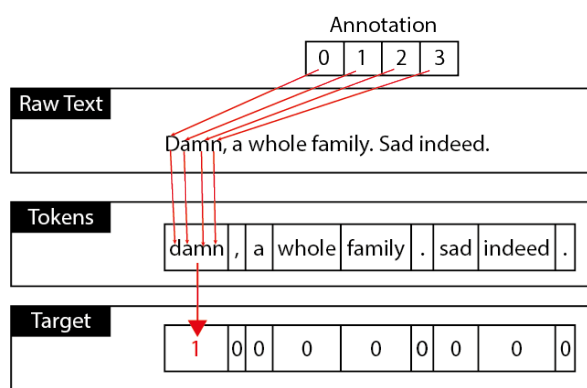


Figure 1: The pipeline of dataset preparation

### 2.1.2 Toxic spans detection

We considered the toxic span detection as a sequence labeling task. Therefore, we fine-tuned SpanBERT pre-trained model on token classification task. First, we tokenize the sentence and map its tokens into indices according to SpanBERT vocabulary. Next, we fed the model with tokens indices. Then, it computes tokens embeddings by applying SpanBERT pre-trained model. After that, we compute the probability if a given token is toxic by applying a linear layer followed by a softmax on tokens embeddings. Finally, the model is trained to optimize the cross-entropy loss. Figure 2 shows the flowchart of the SpanBERT-based model. It is worth noting that we filter predicted spans by removing toxic character offsets that have a size equal to one.

## 2.2 SVM and LIME-based method

### 2.2.1 Data preparation

The data preparation for our unsupervised method can be summarized as follows:

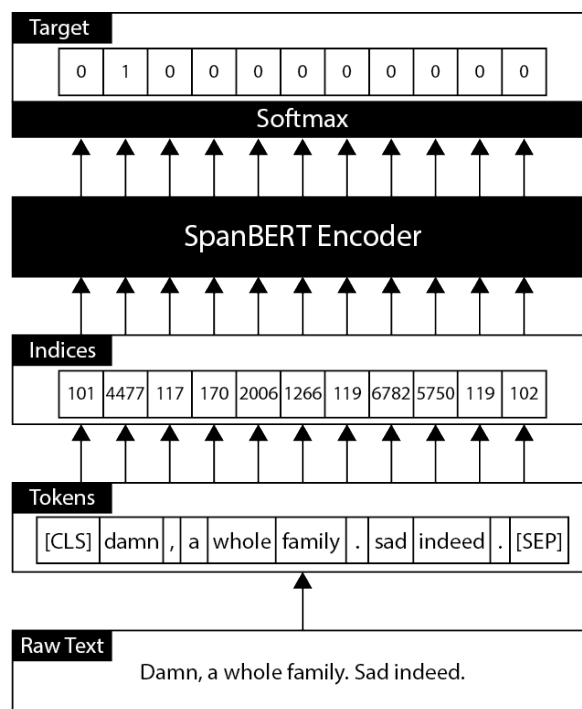


Figure 2: The flowchart of SpanBERT-based model

1. We combine both SemEval 2021 Task 5: Toxic Spans Detection training set which contains 7939 toxic comments, SemEval 2021 Task 5: Toxic Spans Detection test set that contains 2000 toxic comments, and 159571 comments (16225 toxic comments and 143346 non-toxic comments) from Kaggle Jigsaw Toxic comment classification challenge <sup>1</sup> in order to use them for training the linear support vector machine classifier. Later, We label the toxic comments with 1 and non-toxic comments with 0.
2. Word-level uni-grams and bi-grams are extracted, then vectorized using TF-IDF scores.

### 2.2.2 Toxic spans detection

The toxic spans detection task adopted by our unsupervised method can be summarized as follows:

1. We train the linear support vector machine classifier on 26164 toxic comments and 143346 non-toxic comments (the combination of SemEval 2021 Task 5: Toxic Spans Detection training set, SemEval 2021 Task 5: Toxic Spans Detection test set, and a subset of Kaggle Jigsaw Toxic comment classification challenge dataset).

<sup>1</sup><https://raw.githubusercontent.com/iampukar/toxic-comments-classification/master/train.csv>

- We apply the trained model on the SemEval 2021 Task 5: Toxic Spans Detection test set comments to predict their toxicity, then, we use the LIME technique to explain the predictions (Figure 3).
- We discard words that contribute less to the toxic category by applying a thresholding technique. Words with a high influence score, greater or equal to the threshold, are considered toxic, therefore, we retrieve their character offsets (toxic spans).

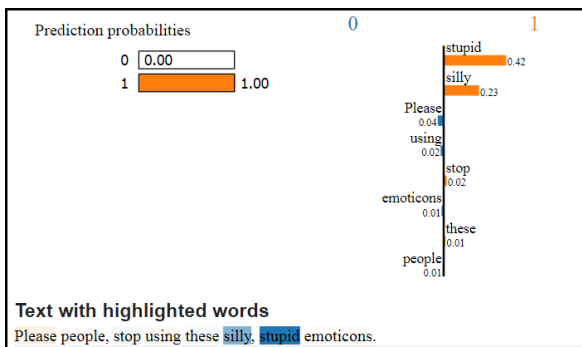


Figure 3: Lime explanations

By training the linear support vector machine classifier on the SemEval 2021 Task 5: Toxic Spans Detection test set, we guarantee that the model accurately predicts the toxicity of its comments with precision, recall, f-score, and accuracy of 1 (the model correctly predict the toxicity of all 2000 reviews in the test set). Besides, we ensure that the LIME explanations are somewhat accurate. In fact, if the model misclassifies the toxicity of the comments, the LIME explanations will be inaccurate since the latter will explain wrong predictions.

From Figure 3, we can see that the words "silly" and "stupid" contribute to the toxic category 42% and 23% respectively in the following toxic comment "Please people, stop using these silly, stupid emoticons". Since we only consider words with high influence scores for the toxic category (greater or equal to 0.13), we keep the two words "silly" and "stupid", and we discard the remaining words. Next, we retrieve their character offsets from the comment as shown in Table 1.

### 3 Experimental results

We experimented our models on the SemEval 2021 Task 5: Toxic Spans Detection dataset. The training set and test set contain 7939 and 2000 toxic

comments labeled with their toxic spans. All our experiments have been conducted in Google Colab environment<sup>2</sup>, The following libraries: Hugging Face<sup>3</sup>, LIME<sup>4</sup>, Scikit-Learn<sup>5</sup>, and PyTorch<sup>6</sup> were used to train and to assess the performance of our models.

#### 3.1 Evaluation Metric

In order to measure the performance of our models, we employ the F1 score proposed in (Da San Martino et al., 2019). Considering a post  $t$  and a system  $A_i$  which predict a set  $S_{A_i}^t$  of toxic character offsets. Let denote by  $G^t$  the expected character offsets. Then, the F1 score of the model  $A_i$  with respect to  $G$  for  $t$  is computed in the following manner:

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|} \quad (1)$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|} \quad (2)$$

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)} \quad (3)$$

where  $|\cdot|$  denotes set cardinality.

#### 3.2 Performance Evaluation

On the one hand, we compared various pre-trained models, including BERT-base, BERT-large, DistilBERT (Sanh et al., 2019), and SpanBERT-large, to compute tokens embeddings. All the models are based on transformers (Vaswani et al., 2017) technique. The SpanBERT model achieves the best results due to the fact that is trained with contiguous masked spans and optimizes the span boundary objective. On the other hand, we compared the logistic regression LIME (LR-LIME) to linear support vector machine LIME (LSVM-LIME). The latter produces superior scores. Table 2 reports the obtained results for both supervised and unsupervised techniques. SpanBERT outperforms all the models by scoring about 0.6783 F1 score. During the fine-tuning of SpanBERT model, we set the hyper-parameters as follows:  $1.5e - 5$  as the learning rate, 3 epochs, 256 as the max sequence length, 4 as batch size, 476 as the warmup steps, and 0.01

<sup>2</sup><https://colab.research.google.com/>

<sup>3</sup><https://huggingface.co/>

<sup>4</sup><https://lime-ml.readthedocs.io/en/latest/lime.html>

<sup>5</sup><https://scikit-learn.org/stable/>

<sup>6</sup><https://pytorch.org/>

Comment	Toxic spans
Please people, stop using these <b>silly</b> , <b>stupid</b> emoticons.	[32, 33, 34, 35, 36, 38, 39, 40, 41, 42, 43, 44]

Table 1: Example of unsupervised toxic spans detection

Method	F1 score
LR-LIME	0.5887938605
LSVM-LIME	0.592141639
DistilBERT	0.6636129383
BERT-base	0.6714433707
BERT-large	0.6677294902
SpanBERT-large	0.6783641122

Table 2: Toxic spans detection results

as the weight decay. For the unsupervised technique, several experiments have been conducted to reach the suitable threshold. Actually, 0.12 and 0.13 thresholds achieved the best performances for LR-LIME and LSVM-LIME, respectively.

## 4 Conclusion

In this paper, we described our models for tackling SEMEval 2021 Task 5: Toxic Spans Detection. Two approaches have been employed. A supervised approach based on transformers technique, where toxic sequences are tokenized and embedded using pre-trained models. We optimize the likelihood of a token to be toxic by minimizing the cross-entropy loss. SpanBERT scored the best results by achieving about 0.6783 F1 score. An unsupervised approach based on shallow machine learning and LIME, which is an explanation technique that explains the prediction of any classifier in an interpretable and faithful manner. Since the top-ranked score was about 0.7083 F1 score, future studies and works will focus on improving the performance of toxic spans detection task.

## References

- Hamza Alami, Said Ouatic El Alaoui, Abdessamad Benlahbib, and Nouredine En-nahnahi. 2020. [LISAC FSDM-USMBA team at SemEval-2020 task 12: Overcoming AraBERT’s pretrain-finetune discrepancy for Arabic offensive language identification](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2080–2085, Barcelona (online). International Committee for Computational Linguistics.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced met-](#)

[rics for measuring unintended bias with real data for text classification](#). In *Companion Proceedings of The 2019 World Wide Web Conference, WWW ’19*, page 491–500, New York, NY, USA. Association for Computing Machinery.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. [SpanBERT: Improving pre-training by representing and predicting spans](#). *Transactions of the Association for Computational Linguistics*, 8:64–77.

John Pavlopoulos, Ion Androutsopoulos, Jeffrey Sorensen, and Léo Laugier. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th International Workshop on Semantic Evaluation*. International Committee for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017a. [Deep learning for user comment moderation](#). In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35, Vancouver, BC, Canada. Association for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017b. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.



- John Pavlopoulos, Nithum Thain, Lucas Dixon, and Ion Androutsopoulos. 2019. [ConvAI at SemEval-2019 task 6: Offensive language identification and categorization with perspective and BERT](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 571–576, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. ["why should i trust you?": Explaining the predictions of any classifier](#). In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, page 1135–1144, New York, NY, USA. Association for Computing Machinery.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC2 Workshop*.
- Anna Schmidt and Michael Wiegand. 2017. [A survey on hate speech detection using natural language processing](#). In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#). In *Proceedings of the 26th International Conference on World Wide Web, WWW '17*, page 1391–1399, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffensEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

# HITMI&T at SemEval-2021 Task 5: Integrating Transformer and CRF for Toxic Spans Detection

Chenyi Wang\*, Tianshu Liu\*, Tiejun Zhao†

Machine Intelligence and Translation Laboratory,  
Harbin Institute of Technology, Harbin, China

wcy708708@126.com, liutianshu99@163.com, tjzhao@hit.edu.cn

## Abstract

This paper introduces our system at SemEval-2021 Task 5: Toxic Spans Detection. The task aims to accurately locate toxic spans within a text. Using BIO tagging scheme, we model the task as a token-level sequence labeling task. Our system uses a single model built on the model of multi-layer bidirectional transformer encoder. And we introduce conditional random field (CRF) to make the model learn the constraints between tags. We use ERNIE as pre-trained model, which is more suitable for the task according to our experiments. In addition, we use adversarial training with the fast gradient method (FGM) to improve the robustness of the system. Our system obtains 69.85% F1 score, ranking 3rd for the official evaluation.

## 1 Introduction

With the prosperity of the Internet, it is easier and easier for people to get information and publish their opinions online. However, sometimes users' opinions can be offensive to others. Because toxic posts will have a negative impact on the network environment, and manual identification is time-consuming and expensive, automatic detection of these behaviors has attracted researchers' attention.

After adapting the hate-speech problem to the problem of word sense disambiguation, an approach to detect hate speech in online text is presented (Warner and Hirschberg, 2012), which uses template-based strategy to generate features and an SVM classifier to identify whether the text is toxic or not. In SemEval-2020 Task 12 (Zampieri et al., 2020) and SemEval-2019 Task 6 (Zampieri et al., 2019), which also related to offensive statements,

transformer-based methods were the most popular approaches for their great advantages in learning word representations in context.

In Semeval-2021 task 5: Toxic Spans Detection (Pavlopoulos et al., 2021), the organizers use posts from the publicly available Civil Comments dataset (Borkan et al., 2019), which already comprises post-level toxicity annotations. After manual annotation, character-level annotation results are obtained, which are the toxic spans we need to locate. The task extends the prior work by identifying spans that make a text toxic, which can better explain why posts are offensive rather than just giving a system-generated unexplained toxicity score.

We model the task as a sequence labeling task because toxic spans are contextually influenced. Our model is in Transformer-CRF architecture, and we try different pre-trained models as the transformer's initialization to fine-tune model suitable for toxic spans detection. The Conditional Random Fields (CRF) (Lafferty et al., 2001) allows the model to learn the constraints between tags. We also use the Fast Gradient Method (FGM) (Miyato et al., 2016) as adversarial training strategy, which applies perturbation to word embedding to enhance the robustness of the model.

The paper is organized as follows: Section 2 briefly introduces the Toxic Spans Detection shared task. Section 3 talks about our system, including pre-processing and post-processing. Section 4 shows our experiment results. Finally, the conclusion and future work are drawn in Section 5.

## 2 Toxic Spans Detection

The research of automatic offensive language detection has gained attention in the past decade. Instead of just classifying the whole comments or documents, the Toxic Spans Detection task requires the system to detect the spans that make a text toxic.

\* Equal contribution.

† Corresponding author.

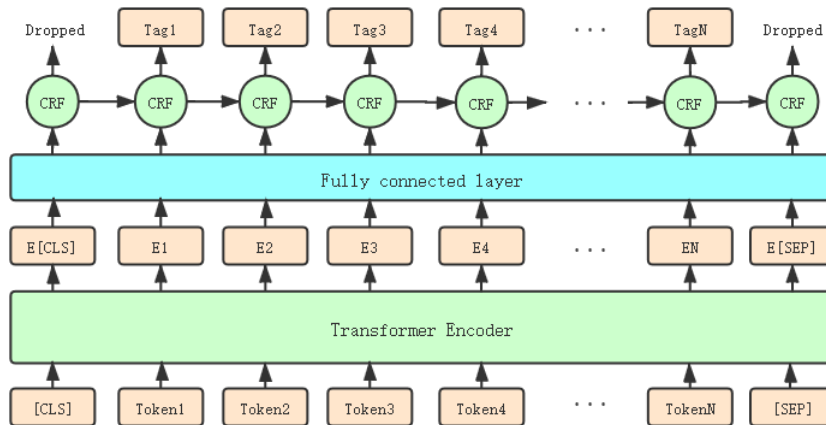


Figure 1: Model structure of Transformer-CRF. In our system, the transformer encoder is ERNIE. “[CLS]” and “[SEP]” are the special input tokens, “TokenN” means the Nth token in tokenized text. “E[CLS]”, “E[SEP]” and “EN” means the output of “[CLS]”, “[SEP]” and the Nth token’s embedding after Transformer Encoder. The output of Transformer-CRF is “TagN”, which is “B-toxic”, “I-toxic” or “O”.

People often judge offensive sentences in terms of words, therefore the toxic spans in this task are always associated with words. In this task, the input sentence may contain no toxic span, which means it is not offensive. On the other hand, there may be more than one word that shows the author’s malice in the sentence. Considering toxic spans are contextually influenced, we model the task as a token-level sequence labeling task and use BIO tagging scheme.

### 3 System description

Our system uses a single model to get the result, which is in Transformer-CRF architecture. In our system, we utilize a pre-trained contextualized language model, namely ERNIE 2.0 (Sun et al., 2019), to identify the toxic tokens. And we introduce CRF to learn the constraints between tags. In addition, we use adversarial training with the FGM to improve our performance.

#### 3.1 Data pre-processing

Toxic spans detection is a character-level task. Considering transformer architecture requires token-level inputs, we choose token-level sequence labeling instead of character-level. When converting character-level sequence labels to token-level, the tokenizer we use is the one used in pre-training. And we lowercase the text before tokenized.

After tokenizing the text, we tag the tokens. If one of the token’s spans is tagged as toxic in the original dataset, considering the tag of the previous token, the token will be tagged as “B-toxic” or “I-toxic” in pre-processed data. If the token’s spans are not tagged as toxic in the original dataset, the token will be tagged as “O” in pre-processed data. Some typical examples are shown in Table 1.

#### 3.2 Transformer-CRF architecture

Our system uses a single model to get token-level predictions. The model is in Transformer-CRF architecture. As shown in Figure 1, it consists of three components: transformer encoder, fully connected layer, and a CRF layer.

First, the transformer encoder is used to extract representations of the input tokens. Based on multi-layer bidirectional transformer encoder, we use ERNIE as pre-trained model to encode. During pre-training, transformer-based language models always use inputs with special tokens (such as [CLS] or [SEP]). Therefore, after tokenizing text into tokens, we insert “[CLS]” at the beginning of the token list and “[SEP]” at the end to make our input closer to what it would be when ERNIE was pre-trained. When we train the model, the tag of “[CLS]” and “[SEP]” is “O”, and we drop the tags of them during predicting.

After we get the embeddings of tokens, the representations are fed into a fully connected layer to

Origin tags	Origin data	Tokens and tags
8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, ...	Another violent and aggressive immigrant killing a innocent and intelligent US Citizen....	another O violent B-toxic and I-toxic aggressive I-toxic ... ..
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17	What a knucklehead. How can anyone not know this would be offensive??	what O a O kn B-toxic ##uck I-toxic ##le I-toxic ##head I-toxic ... ..

Table 1: Typical Examples, where “Origin tags” means the toxic spans given in the dataset, “Origin data” means the comment given in the dataset, “Tokens and tags” means the tokens and tags after tokenized and tagged.

reduce dimension. Finally, the CRF layer decodes the reduced representations into the most probable tag sequence using the Viterbi algorithm. Using only the fully connected layer may output illegal tags (e.g. “... O I-toxic O ...”), while introducing the CRF layer allows the model to learn the constraints between tags. When fine-tuning the pre-trained model, the layer closer to the input is more likely to learn more simple features. We want to modify the deeper weights more to adapt to the target task. Therefore, we use different learning rates for different parts of the network. The transformer encoder has a lower learning rate, while the fully connected layer and the CRF layer have a higher learning rate.

We train the model maximizing the log-likelihood of the given sequence of tags, as compared to the gold training labels. Except that, we use the FGM as an adversarial training strategy to improve the performance of our system. FGM applies perturbation to word embedding to enhance the robustness of the model.

Given a token sequence  $S = t_1, t_2, \dots, t_N$  as input where  $N$  denotes the sequence length. The process for the model to obtain token-level results is as follows:

$$e_i = \text{transformer\_encoder}(t_i) \quad (1)$$

$$\text{out}_i = W_o e_i + b_o \quad (2)$$

$$\text{tag}_i = \text{crf}(\text{out}_i) \quad (3)$$

Where  $t_i$  is the current token, and  $e_i$  denotes the output of transformer encoder. Then  $e_i$  is fed into

fully connected layer to reshape into  $\text{out}_i$ . After that, the CRF layer use  $\text{out}_i$  to get token-level result  $\text{tag}_i$ .

### 3.3 Post-Processing

Our model produces token-level predictions, but detecting toxic spans within the text is a character-level task. To get the results, we use the opposite way of pre-processing to get spans from model’s predictions. Particularly, we assume the blank between toxic tokens should be tagged as toxic, too. This is observed from the spans of the dataset.

## 4 Experiment

### 4.1 Dataset

We trained our models by SemEval-2021 Task 5 training data which consists of 7,939 samples with a total of 139,115 toxic spans. We convert the span from character-level to token-level, as described in section 3.1. Then we get a total of 31,114 toxic tokens, with an average of 3.92 per sample. And each sample contains an average of 48.50 tokens.

### 4.2 Metric

Let system  $A$  return a set  $S_A^t$  of character offsets, for the post  $t$  that found to be toxic. Let  $S_G^t$  be the set of character offsets of the ground truth annotations of  $t$ . We compute the F1 score of system  $A$  with respect to the ground truth  $G$  for post  $t$  as follows, where  $|\cdot|$  denotes set cardinality.

$$P^t = \frac{|S_A^t \cap S_G^t|}{|S_A^t|} \quad (4)$$

Method	F1
BERT-CRF	0.6933
ELECTRA-CRF	0.6944
ERNIE-CRF	<b>0.6985</b>
ERNIE-CRF w/o-adv	0.6964
ERNIE-CRF nltk-preprocessing	0.6557

Table 2: Results on the official evaluation testing data. “w/o-adv” means no adversarial training. “nltk-preprocessing” means using NLTK for pre-processing.

$$R^t = \frac{|S_A^t \cap S_G^t|}{|S_G^t|} \quad (5)$$

$$F_1^t = \frac{2 \cdot P^t \cdot R^t}{P^t + R^t} \quad (6)$$

If  $S_G^t$  is empty for some post  $t$  (no gold spans are given for  $t$ ), we set  $F_1^t = 1$  if  $S_A^t$  is also empty, and  $F_1^t = 0$  otherwise. We finally average  $F_1^t$  over all the posts  $t$  of an evaluation dataset  $T$  to obtain a single  $F_1$  score for system  $A$ .

$$F_1 = \frac{\sum_{t \in T} F_1^t}{|T|} \quad (7)$$

### 4.3 Experiment Settings

We try different pre-trained model as the transformer’s initialization such as BERT (Devlin et al., 2018), ELECTRA discriminator (Clark et al., 2020) and ERNIE 2.0 (Sun et al., 2019). We find that the models initialized with ERNIE 2.0 always achieve better performance. So we select ERNIE 2.0 as the transformer’s initialization, which has 768 hidden units, 12 heads, 12 hidden layers.

For other parameters, we use streams of 256 tokens, a mini-batch of size 32, transformer’s learning rate of  $3e-5$ , CRF’s learning rate of  $1e-3$ , the epoch of 3, and the random seed of 42.

### 4.4 Testing Results

As shown in Table 2, we build five systems including: (1) **BERT-CRF** means model using BERT (Devlin et al., 2018) as the transformer’s initialization; (2) **ELECTRA-CRF** means model using ELECTRA discriminator (Clark et al., 2020) as the transformer’s initialization; (3) **ERNIE-CRF** means model using ERNIE 2.0 (Sun et al., 2019) as the transformer’s initialization; (4) **ERNIE-CRF w/o-adv**; (5) **ERNIE-CRF nltk-preprocessing**. All models are Transformer-CRF architecture. All models use adversarial training with the FGM except **ERNIE-CRF w/o-adv**. All models convert character-level sequence labels to token-level with

tokenizers used in pretraining, except for **ERNIE-CRF nltk-preprocessing**, which uses NLTK (Bird et al., 2009) for pre-processing.

Table 2 shows the overall performances of our models on the official evaluation testing data. The ERNIE based model achieves better performance than both the BERT based model and the ELECTRA based model. We conjecture that ERNIE 2.0 is pre-trained through multi-task learning, which allows it to capture lexical, syntactic, and semantic information, such as named entities and discourse relations. And this makes it more suitable for Toxic Spans Detection task.

Adversarial training proved to be effective. Although only a small improvement has been achieved, experiments show that FGM can always achieve a stable improvement. Adversarial training adds some perturbation to the input, which makes the model more robust and has a better performance on the unknown test set.

**ERNIE-CRF** achieves more than 4 points improvements over **ERNIE-CRF nltk-preprocessing**, which proves the importance of choosing the correct method to convert character-level sequence labels to token-level. We conjecture that there is a mismatch between the word segmentation results of NLTK and the input required by the Transformer model.

### 4.5 Attempts with no obvious improvement

It is worth mentioning that the best performance of our system is based on a single model. We tried model ensemble to improve the performance of the single model but failed. Due to the limitation of time and submission, we did not find an ensemble method with obvious improvement.

The BiLSTM network has a strong ability to capture long-term dependencies of the input sequence for sequence labeling task (Huang et al., 2015). We tried to add BiLSTM layer after transformer. But this resulted in overfitting, and the training speed

was greatly reduced.

We also tried to combine the information from word embedding with the information from the deep layer. It is proved that, in the machine translation task, the low layers of the network are more focused on lexical information, while deeper layers pay more attention to word meaning (Belinkov et al., 2017). In toxic spans detection task, we consider the information from the lower layers to be important. So we concatenate the word embedding layer with the output of ERNIE, but this didn't work.

## 5 Conclusion and Future Work

The paper describes our system at SemEval-2021 Task 5, which integrating Transformer and CRF for Toxic Spans Detection task. It is shown that, in this task, using ERNIE as pre-trained model achieves better performance in Transformer-CRF architecture. We convert the character-level sequence labeling task into token-level, and prove the importance of preprocessing method. We also use adversarial training with the FGM to improve the robustness of the system. Our system achieves the third F1 score in official evaluation.

Since the best performance of our system is based on a single model, we are planning to find an effective ensemble method to improve the performance of the single model in the future.

## References

- Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.
- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc."
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.
- William Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web. In *Proceedings of the second workshop on language in social media*, pages 19–26.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. Semeval-2020 task 12: Multilingual offensive language identification in social media (offenseval 2020). *arXiv preprint arXiv:2006.07235*.

# AStarTwice at SemEval-2021 Task 5: Toxic Span Detection using RoBERTa-CRF, Domain Specific Pre-Training and Self-Training

Thakur Ashutosh Suman\*

Abhinav Jain\*

Indian Institute of Technology (Indian School of Mines) Dhanbad, India

{ashutoshsuman99, jain.abhinav02}@gmail.com

## Abstract

This paper describes our contribution to SemEval-2021 Task 5: Toxic Spans Detection. Our solution is built upon RoBERTa language model and Conditional Random Fields (CRF). We pre-trained RoBERTa on Civil Comments dataset, enabling it to create better contextual representation for this task. We also employed the semi-supervised learning technique of self-training, which allowed us to extend our training dataset. In addition to these, we also identified some pre-processing steps that significantly improved our F1 score. Our proposed system achieved a rank of 41 with an F1 score of 66.16%.

## 1 Introduction

In recent years there has been an exponential increase in the use of social network platforms. With rising abusive language and hate on such platforms, it is more important than ever to maintain online conversations constructive and inclusive. This problem can be tackled by filtering toxic comments/posts. The massive volume of data generated at a fast pace makes manually filtering each comment complicated and time-consuming. This process can be automated by modelling it as a supervised classification problem. A similar task was proposed in SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media (OffensEval) (Zampieri et al., 2019). Most of the top-ranked teams in this task used transformer language models (Liu et al., 2019a; Zhu et al., 2019; Pelicon et al., 2019; Wu et al., 2019) or an ensemble of CNN and RNN (Mahata et al., 2019; Mitrović et al., 2019) to classify the sentences.

The problem with the above approach is that it doesn't give moderators much knowledge about the reason for a sentence's toxicity. Highlighting

Equal Contribution. Author order determined by a coin flip

toxic spans can help human moderators who frequently deal with long comments and prefer attribution rather than just an unexplained toxicity score. SemEval 2021 Task 5: Toxic Span Detection (Pavlopoulos et al., 2021) gave a chance to propose NLP systems to solve this problem. The task is concerned with developing systems that can recognise spans that contribute to the text's toxicity.

This task had a few challenges. Since the samples were from an online commenting platform, they were grammatically incorrect and consisted of many out of vocabulary words. The noisy and ambiguous structure of comments significantly hampers the performance of general NLP models. The training dataset had a little less than 8000 samples. Thus, there was a need to select systems that can produce meaningful results, even with a limited number of training samples. Undoubtedly, the hardest part is to identify spans that can account for the toxicity of the sample. The span could be as small as a single token and as large as the sample itself. The linguistic variations in the usage of words and phrases make such attribution even more difficult.

We formulated the task as a sequence tagging problem and used RoBERTa (Liu et al., 2019b), a pre-trained Transformer-based (Vaswani et al., 2017) language model as our base model. We further pre-trained RoBERTa on the Civil Comments Dataset as a masked language model (Devlin et al., 2018) to create a domain-specific model. We employed a Conditional Random Field (CRF) layer (Lafferty et al., 2001) for predicting the most probabilistic sequence of labels for each input sequence. We also applied a few pre-processing steps, which lead to significant performance improvements. Lastly, we leveraged the semi-supervised learning technique of self-training (Yarowsky, 1995; Liao and Veeramachaneni, 2009; Jurkiewicz et al., 2020) by training our model on the manually annotated

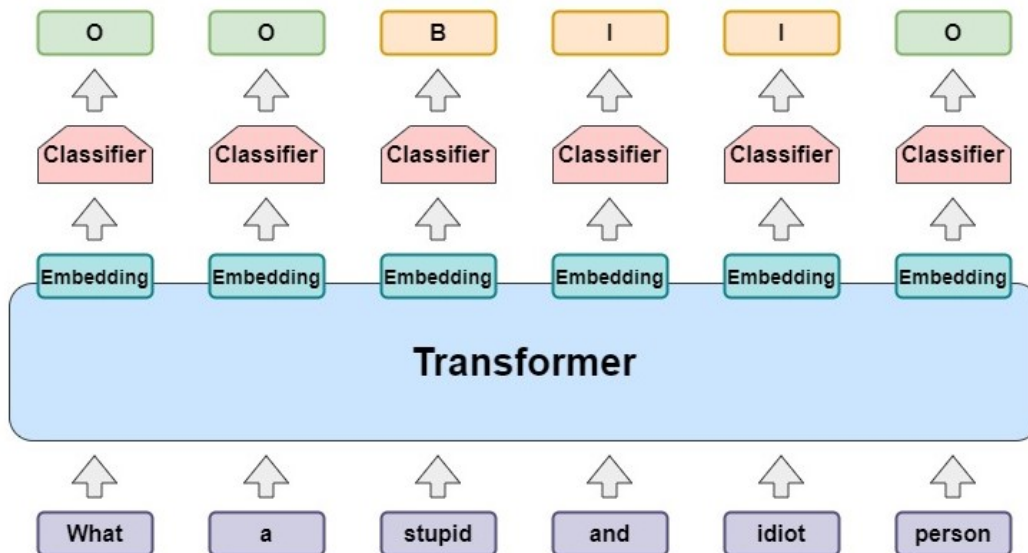


Figure 1: Our Model Architecture. We used RoBERTa as our transformer. Classifier constitutes two dense layers and a CRF layer with three labels.

dataset and using it to further extend the training set by generating toxic spans for other unannotated datasets. We have made our system’s implementation available through GitHub<sup>1</sup>.

The rest of the paper is organised as follows. Section 2 explains our model implementation in detail. Section 3 and 4 presents our experimental setup and achieved results, respectively. In section 4, we perform error analysis, followed by conclusions in the last section.

## 2 System Description

### 2.1 Pre-Training

Toxic comments have a different language construct from the general language. Their slang and obfuscated content (van Aken et al., 2018) make it difficult for the language models pre-trained on broader datasets to understand them. Similar to other domain-specific models (Beltagy et al., 2019; Lee et al., 2020; Paraschiv et al., 2020), we pre-trained the RoBERTa-base model on the Civil comments dataset using Masked Language Modelling (MLM) (Devlin et al., 2018) to provide the necessary domain knowledge and created our model RoBERTa(p). The original weights of RoBERTa-base served as the starting point for the pre-training. The pre-training was done for 0.2 million steps with a batch size of 32 and a learning rate of  $2e-5$ .

<sup>1</sup>[https://github.com/jain-abhinav02/Toxic\\_Spans\\_Detection](https://github.com/jain-abhinav02/Toxic_Spans_Detection)

### 2.2 Fine-Tuning

We formulated the task as a token level sequence tagging problem where we classify each token as Begin, Inside or Outside (BIO scheme). Having begin and end tags helps formulate the notion of spans better and creates dependencies between various tokens of a toxic span (Singh et al., 2020), allowing it to perform better than other alternatives such as IO (Inside Outside).

**Pre-Processing:** We applied a few pre-processing steps before fine-tuning RoBERTa on the input text samples. First, we converted all the text samples to lowercase. We observed that punctuation marks did not add any significant information to the semantics of a sentence. Therefore, as a part of the data cleaning, punctuation marks such as commas and dashes were removed. We also collapsed multiple space characters into a single space.

**Model:** We provided the text samples as input to our pre-trained RoBERTa(p) model to get 768-dimensional contextual embeddings for each token. These contextual embeddings were passed through two dense layers of 512 and 128 dimensions, followed by a Conditional Random Fields (CRF) (Lafferty et al., 2001) layer with three labels (B-Begin, I-Inside or O-Outside). The CRF layer models the correlation between the labels predicted for the individual tokens. It receives the logits for each input token and predicts the most probabilistic sequence



Model	Tag	F1	Precision	Recall
RoBERTa	IO	0.6091	0.5831	0.7224
RoBERTa(p)	IO	0.6183	0.5841	<b>0.7408</b>
RoBERTa(p) + PP	IO	0.6376	0.6259	0.7264
RoBERTa(p) + PP + CRF	IO	0.6422	0.6323	0.7246
RoBERTa(p) + PP + CRF	BIO	0.6566	0.6512	0.7203
RoBERTa(p) + PP + CRF + ST(1)	BIO	0.6613	0.6537	0.7295
RoBERTa(p) + PP + CRF + ST(2)	BIO	<b>0.6634</b>	<b>0.6590</b>	0.7262

Table 1: Our model results on Test Set. RoBERTa(p) is our model pre-trained on domain-specific data. PP stands for Pre-processing. ST(1) and ST(2) represents self-training first and second iteration results, respectively.

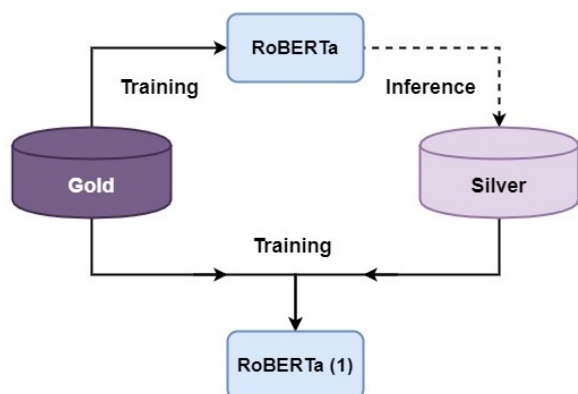


Figure 2: Self-Training of RoBERTa

of labels for each input sequence. Figure 1 shows our model architecture.

**Post-Processing:** The tokens decoded as B-Begin or I-Inside were marked as toxic. The character spans corresponding to these toxic tokens were added to the predicted spans. Two consecutive spans were merged if separated by at most five characters, provided all of them are non-alphabetic.

### 2.3 Self-Training

The best performing model on the manually annotated dataset (gold dataset) was used to generate toxic spans for the unannotated dataset. When selecting the unannotated data, we followed the process similar to the one used for creating the gold dataset (Pavlopoulos et al., 2021) that is, filter the most toxic samples (toxicity  $\geq 0.80$ ) from the Civil Comments dataset and select a random set of 10,000 samples. This process allowed the silver data to have similar toxicity distribution as the gold data. The newly generated annotations (silver dataset) were then used along with the gold dataset to train a new model. The model trained on the combined gold and silver dataset gave better performance (F1 score: 66.13%) than the one trained

only on the gold dataset (F1 score: 65.66%). We repeated this process for one more iteration with another random set of 10,000 samples (F1 score: 66.34%). Figure 2 gives a simplistic idea of self-training.

## 3 Experimental Setup

**Data:** Each training example consisted of a text sample in English, and its ground truth toxic span provided as a list of character offsets (possibly empty). The posts were sampled from the publicly available Civil Comments dataset. The training set consisted of 7939 samples. We randomly sampled 20% of it as the development set. The test set for the evaluation phase had 2000 samples. In the training dataset, sample length varies from 1 to 421 tokens, with an average length of 47 tokens when tokenized using the RoBERTa-base tokenizer. Nearly 10% of all tokens in the training dataset are marked as toxic. The mean span length is 17.5 characters and 485 samples have empty spans. Further details about the dataset can be found in the task description paper (Pavlopoulos et al., 2021).

**Evaluation measures:** The evaluation measure for a sample is the F1 score between the predicted spans and the ground truth spans as defined in the SemEval-2021 Task 5 paper (Pavlopoulos et al., 2021). The overall score is obtained by taking the mean of the F1 score of all samples in the test set.

**Implementation details:** Since the text samples were lengthy, we split them into sentences. We used spacy’s sentence tokenizer for this purpose. Adam (Kingma and Ba, 2014) was used as the optimizer with a learning rate of  $1e-5$ . The batch size was set to 32. All the model variants were trained for at most five epochs. We used the RoBERTa-base tokenizer and model implementation provided

Original Spans	Predicted Spans
<i>Small Sentences</i>	
[Sick pervert]	Sick pervert
[bloody wankers]	bloody wankers
<i>Rare Words</i>	
Just another day in the Trump <b>[kleptocracy]</b> .	Just another day in the Trump kleptocracy.
Should ha taken his weapon n <b>[pulverized]</b> him	Should ha taken his weapon n pulverized him
<i>Non-swear words</i>	
When you're ready to place a monetary bet on whether these guys get charged with terrorism, lemme know. Otherwise, I know you're <b>[all bluster and no balls]</b> .	When you're ready to place a monetary bet on whether these guys get charged with terrorism, lemme know. Otherwise, I know you're all bluster and no <b>[balls]</b> .
Yes, I agree. We need an electoral college for Washington so that Seattle no longer votes for Washington. My goodness,they are a <b>[bunch of loons]</b> !	Yes, I agree. We need an electoral college for Washington so that Seattle no longer votes for Washington. My goodness,they are a bunch of <b>[loons]</b> !

Table 2: Common errors made by our model. Spans are made bold and are present in between square brackets.

by Huggingface<sup>2</sup>. The RoBERTa model was followed by two dense layers with 512 and 128 units with relu (Agarap, 2018) as the activation function and a dropout rate of 0.1. The output layer had two or three labels depending on the tagging scheme. We applied the post-processing steps mentioned in section 2.2 for all the model variants.

## 4 Results

Table 1 shows that our RoBERTa(p) model outperforms the original RoBERTa model. As suggested earlier, domain-specific pre-training allows the model to understand the language construct of toxic comments better. Additionally, we observe a significant increase in performance by adding pre-processing steps as it makes the model more robust to the noise present in the text samples. Adding the CRF layer further improves the F1 score by eliminating the problem of independent label prediction. It is evident from table 1 that the BIO tagging scheme performs better than the IO tagging scheme when working with CRF, suggesting it can better understand the span nature of the output. Finally, using two rounds of self-training helped us achieve our best F1 score, 66.34%<sup>3</sup>.

One interesting observation that can be drawn from Table 1 is that for almost all the models, the

<sup>2</sup><https://github.com/huggingface/transformers>

<sup>3</sup>We achieved an F1 score of 66.16% in the official competition. However, our model achieved a even higher F1 score 66.34%, when the predictions of a different epoch were used for evaluation.

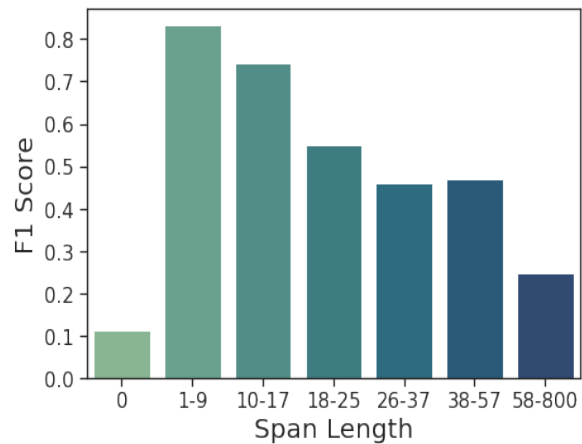


Figure 3: Distribution of F1 Score across different span lengths. Here span length refers to the total length of the toxic span in each sample. The value represented is the mean F1 score of all the text samples whose toxic span length falls in a particular range.

recall remains constant and improvement in F1 is due to improvement in precision. The constancy of recall indicates that few spans are not captured as toxic by any of the models.

## 5 Error Analysis

Figure 3 shows the variation of the F1 score across different toxic span lengths on the test dataset. Our model achieved a very high F1 score when one (Span Length 1-9, Mean F1 Score: 83.17%) or two (Span Length 10-17, Mean F1 Score: 74.44%) words are marked as toxic in a text sample. As the number of characters marked as toxic increases,

the F1 score falls drastically, reaching as low as 24.82% when more than 58 characters are marked as toxic. There are two main reasons for this. First, it is easier for the model to capture short-term dependencies than long-term dependencies. Second, only 10% of the training data has a span length of more than 25 characters making the model less equipped to capture such toxic spans.

To investigate our model’s most problematic cases, we analysed the samples for which our model gave a zero F1 score. There were 447 such samples, of which 349 samples did not have any toxic span in the ground truth. This is also reflected in Figure 3, as the mean F1 score of all the samples with zero span length is 11.42%. Further analysis revealed that our model tends to mark those tokens as toxic, which were frequently found to be toxic elsewhere. A few samples with empty toxic spans had doubtful gold annotations. However, in other samples, our model failed to capture the sentence’s context precisely and predicts tokens that were not used in a toxic sense.

Table 2 shows other standard errors our model makes. It seems that our model has a problem with small sentences. More often than not, it misses the toxic span present in it and returns an empty span. A similar case occurs when it encounters text samples with rare toxic words. These words may be present in very few examples or be completely absent from the training dataset, making our model less endowed to understand them. Other than these, our model sometimes misses the non-swear words in a toxic span.

## 6 Conclusion

This paper described our system developed for SemEval-2021 Task 5: Toxic Span Detection. We built our solution on the RoBERTa language model and Conditional Random Fields (CRF). Though RoBERTa alone can achieve great results, we highlighted the benefits of using external datasets and the performance improvements it can help us achieve. We pre-trained RoBERTa on the Civil Comments dataset to impart domain-specific knowledge to it. We also employed the semi-supervised learning technique of self-training to extend our training dataset. In addition to these, we also discovered some pre-processing steps that significantly improved our F1 score. Experimenting with different tagging schemes, we found out that the BIO scheme works the best with CRF.

In future, we plan to experiment with other language models such as T5 (Raffel et al., 2019), XLNet (Yang et al., 2019) and DeBERTa (He et al., 2020). The system could also benefit from the addition of syntactic and semantic features at the word and sentence level.

## References

- Abien Fred Agarap. 2018. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. *arXiv preprint arXiv:1809.07572*.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. Applicaai at semeval-2020 task 11: On roberta-crf, span cls and whether self-training helps them. *arXiv preprint arXiv:2005.07934*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML ’01*, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Wenhui Liao and Sriharsha Veeramachaneni. 2009. A simple semi-supervised algorithm for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, pages 58–65, Boulder, Colorado. Association for Computational Linguistics.

- Ping Liu, Wen Li, and Liang Zou. 2019a. [NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Debanjan Mahata, Haimin Zhang, Karan Uppal, Yaman Kumar, Rajiv Ratn Shah, Simra Shahid, Laiba Mehnaz, and Sarthak Anand. 2019. [MIDAS at SemEval-2019 task 6: Identifying offensive posts and targeted offense from Twitter](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 683–690, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Jelena Mitrović, Bastian Birkeneder, and Michael Granitzer. 2019. [nlpUP at SemEval-2019 task 6: A deep neural language model for offensive language detection](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 722–726, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Andrei Paraschiv, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. Upb at semeval-2020 task 11: Propaganda detection with domain-specific trained bert. *arXiv preprint arXiv:2009.05289*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Andraž Pelicon, Matej Martinc, and Petra Kralj Novak. 2019. [Embeddia at SemEval-2019 task 6: Detecting hate with neural network and transfer learning approaches](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 604–610, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Paramansh Singh, Siraj Sandhu, Subham Kumar, and Ashutosh Modi. 2020. newssweeper at semeval-2020 task 11: Context-aware rich feature representations for propaganda classification. *arXiv preprint arXiv:2007.10827*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Zhenghao Wu, Hao Zheng, Jianming Wang, Weifeng Su, and Jefferson Fong. 2019. [BNU-HKBU UIC NLP team 2 at SemEval-2019 task 6: Detecting offensive language using BERT model](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 551–555, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- David Yarowsky. 1995. [Unsupervised word sense disambiguation rivaling supervised methods](#). In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.
- Jian Zhu, Zuoyu Tian, and Sandra Kübler. 2019. Um-iu@ ling at semeval-2019 task 6: Identifying offensive tweets using bert and svms. *arXiv preprint arXiv:1904.03450*.

# NLP\_UIOWA at Semeval-2021 Task 5: Transferring Toxic Sets to Tag Toxic Spans

**Jonathan Rusert**

Department of Computer Science

University of Iowa

Iowa City, IA, USA

jonathan-rusert@uiowa.edu

## Abstract

We leverage a BLSTM with attention to identify toxic spans in texts. We explore different dimensions which affect the model’s performance. The first dimension explored is the dataset the model is trained on. Besides the provided dataset, we explore the transferability of 5 different toxic related sets, including offensive, toxic, abusive, and hate sets. We find that the solely offensive set shows the highest promise of transferability. The second dimension we explore is methodology, including leveraging attention, employing a greedy remove method, using a frequency ratio, and examining hybrid combinations of multiple methods. We conduct an error analysis to examine which types of toxic spans were missed and which were wrongly inferred as toxic along with the main reasons why they occurred. Finally, we extend our method via ensembles, which achieves our highest F1 score of 55.1.

## 1 Introduction

Social media platforms (e.g. Twitter) benefit from a welcoming environment. Toxicity introduced by users can drive new (or existing) users away. Therefore, it is beneficial for platforms to moderate the amount of toxic language allowed. Often human moderators are employed to address this. However, as the numbers of users increase on a platform, it can be difficult for human moderation as the sheer amount of content generated increases as well. Thus platforms employ automated assistance (Terdiman, 2018).

While automated toxic language classifiers have made strides at determining if text is toxic (Yenala et al., 2018; Zhang et al., 2018), much work is still left for humans who are reviewing the text. In large texts, the reason for toxicity might only be one or two words, however, the moderator will have to

Text	Toxic Span
What a jerk!	[7, 8, 9, 10]
“Trump is up to the task” LMAO, you are an idiot	[44, 45, 46, 47, 48]
Canada: Institution- alised mediocrity.	[ ]
Contrary to popular be- lief not all criminals are stupid. Fortunately, there are plenty like this moron that are.	[49, 50, 51, 52, 53, 54, 98, 99, 100, 101, 102]

Table 1: Examples of toxic spans in toxic texts. The numbers indicate the character position of the toxic span. Empty brackets indicate no specific span.

scan the entire text to find the toxicity. Repeating this process across many texts is cumbersome. Automated systems could assist human moderators further by noting which areas in the text are toxic, thus allowing the human to quickly check text for toxicity. Finding these toxic areas is the task that is proposed by (Pavlopoulos et al., 2021). Specifically, given English toxic text, determine where the “toxic spans” occur, where a “toxic span” is the character positions in the text where the toxicity appears. For example, in “your an idiot, this is a tax based on a lie”, the toxic span would be [8, 9, 10, 11, 12] or the character positions of “idiot”. Note that not all toxic texts have toxic spans. For example, “Religious people no longer have a place in this country.” is a toxic phrase, yet no specific words are themselves toxic. More examples can be found in Table 1.

To find toxic spans in text, we employ a Bidirectional Long Short-term Memory model (BLSTM) with attention. We examine several indirect methods which align with the BLSTM, including using the attention layer for information. While the provided dataset allows for direct methods, i.e. meth-

ods which directly use the given spans for training, this span format is new to this task. Thus many previously developed toxic datasets would need to be converted to be useful to a direct model (since they are binary labeled sets, not span labeled sets). However, the indirect methods we explore can leverage previously developed datasets, which opens up research possibilities. Therefore, we examine how well toxic texts other than the provided training data transfer to this task. An additional benefit is that since toxicity can be subjective (Aroyo et al., 2019), examining how different datasets transfer to this task will help illuminate the similarities and differences between definitions of toxicity. Finally, we provide our code and outputs for reproducibility<sup>1</sup>.

## 2 Approach

We expand on the approach we use to identify toxic spans. We examine both different toxic based datasets as well as different BLSTM based methods.

### 2.1 Base Architecture

We leverage a bidirectional LSTM (BLSTM) with attention as our base architecture. The BLSTM has a hidden layer size of 200. We use glove embeddings<sup>2</sup> (Pennington et al., 2014) of size 200 trained on Wikipedia and Gigaword corpora. We train our system over 10 epochs with a batch size of 64.

### 2.2 Toxic Datasets

We examine the provided training set as well 5 different toxic-based data sets. All are in English to match the provided test set. Each additional dataset is explicitly toxic or in a related area (e.g. offense). The dataset chosen can affect performance on the test data as data can vary in their definitions of toxicity. The datasets examined are:

**Toxic Train:** The training set provided by the task organizers. This set had no non-toxic examples thus we pulled non-toxic examples from the Kaggle Toxic set at a 1-to-1 ratio. This results in a training set size of 15,878.

**OLID:** (Zampieri et al., 2019a) A dataset composed of offensive and non-offensive tweets. It was used in both OffensEval 2019 (Zampieri et al., 2019b) and OffensEval 2020 (Zampieri et al., 2020)

(offense language classification tasks). The training set is composed of 13,240 tweets.

**Kaggle Toxic:** Comments from Wikipedia’s talk page edits, presented in the Kaggle toxic classification challenge<sup>3</sup>, labeled as toxic or not toxic (as well as further divided into subcategories). The training set is 159,571 comments.

**Founta:** (Founta et al., 2018) A dataset of tweets annotated via the CrowdFlower platform. The labels are hateful, abusive, or none. For training, we combine hateful and abusive into a single toxic category. The training set is of size 85,966.

**Davidson:** (Davidson et al., 2017) Annotated tweet dataset composed of offensive, hatespeech, and neither tweets. Our training set combines offense and hate into a single toxic label and is of size 24,783.

**Gab Hate:** (Kennedy et al., 2018) A dataset from Gab social network<sup>4</sup>. Each post is annotated by at least 3 annotators as either hateful or not. Similar to prior research, our positive (toxic) labeled are those for which the majority of annotators agree. The training set is of size 27,665.

### 2.3 Methods

We leverage previously examined methods in areas such as robustness (Hsieh et al., 2019) and style transfer (Xu et al., 2018; Wu et al., 2019). Each method stems from the BLSTM, but some can be adapted for other machine learning systems.

**Attention:** Previous work has used attention for robustness attacks (Hsieh et al., 2019) and style token masking (Xu et al., 2018). Following these, we leverage the attention layer in the BLSTM. Each token (word) has an attention score used to generate the class label (toxic, non-toxic). We use this score to determine if a given token is toxic. We label each token with an attention score of greater than the average attention score as part of the toxic span.

**Frequency Ratio:** In theory, a toxic token should exist more frequency in toxic texts, compared to non-toxic texts. Following previous research (Wu et al., 2019), we generate a frequency score for each token:

$$s_c(u, a) = \frac{\text{count}(u, D_a) + \lambda}{(\sum_{a' \in A, a' \neq a} \text{count}(u, D_{a'})) + \lambda} \quad (1)$$

where  $u$  is the token,  $a$  is an attribute (e.g. toxic or non-toxic),  $D_a$  represents the texts with attribute  $a$ ,

<sup>1</sup>Code: <https://github.com/JonRusert/semeval-2021-task-5>

<sup>2</sup><https://nlp.stanford.edu/projects/glove/>

<sup>3</sup><https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge>

<sup>4</sup><https://gab.com/>

		Toxic Datasets						Average
		Train	OLID	Kaggle-Toxic	Founta	Gab	Davidson	
Methods	Attention	50.1	50.1*	39.4	35.2	17.4	22.8	35.8
	Greedy Remove	38.3	44.7	43.3	41.9	20.8	18.6	34.6
	Frequency Ratio	41.5	45.5	22.5	44.3	19.7	8.0	30.3
	Simple Hybrid	38.4	45.5	41.7	39.6	17.7	21.7	34.1
	Recall Hybrid	37.9	45.2	38.4	34.6	17.8	22.7	32.8
	Precision Hybrid	<b>50.7</b>	49.8	44.6	42.6	20.8	19.0	<b>37.9</b>
	Average	42.8	<b>46.8</b>	38.3	39.7	19.0	18.8	
	Top System	70.8						

Table 2: Full Results on the provided Toxic Span Test Set. Results are F1 scores. \* - Version submitted to task organizers. Top System is the highest result submitted by another team to the task organizers.

and  $\text{count}(u, D_a)$  represents the number of times  $u$  appears in  $D_a$ . The score ( $s_c(u, a)$ ) is then measured against a threshold to determine whether the token is representative of that attribute. For our method, we chose a threshold of 5, such that, if  $s_c(u, a) \geq 5$ , then  $u$  is considered part of the toxic span.

**Greedy Remove:** Also seen as “Greedy select” (Hsieh et al., 2019) or “Important Score” (Lee, 2020) in previous research, greedy remove determines the amount each token is contributing to the text by removing each token one-by-one (each token is replaced after drop is noted) and noting the drop in toxic probability. The token with the highest probability is removed (i.e. added to the toxic span) and then the BLSTM classification is checked. If the text is no longer considered toxic, the toxic span is returned, otherwise the next token with the highest drop is removed and it is added to the toxic span. This process is continued until the text is no longer considered toxic by the BLSTM.

**Simple Hybrid:** We combine the greedy remove and attention approaches. We first obtain the spans from the greedy remove. If the span is empty (i.e. “[ ]”), then we choose the spans found by the attention method, otherwise we choose the spans obtained from the greedy remove method.

**Recall Hybrid:** Similar to the simple hybrid, however, we aim to maximize recall. We obtain spans from both the greedy remove and attention approaches and return the union of the two sets.

**Precision Hybrid:** Similar to the simple hybrid, however, we aim to maximize precision. We obtain spans from both the greedy remove and attention approaches and return the intersection of the two sets.

### 3 Experimental Results

#### 3.1 Experiments

We train corresponding BLSTMs on each of the 6 presented datasets (Section 2.2). We test the methods of each BLSTM on the provided toxic test set. The test set contains 3,180 texts. The score used is the F1 for a given text  $t$ , where  $A_i$  is the system and  $G$  is ground truth, and  $S^t$  is the set of toxic spans:

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)} \quad (2)$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|} \quad (3)$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|} \quad (4)$$

When  $S_G^t$  is empty and  $S_{A_i}^t$  is empty, then  $F_1^t(A_i, G) = 1$ , otherwise  $F_1^t(A_i, G) = 0$ . Finally, to obtain a single score, the F1’s are averaged across texts.

#### 3.2 Results

We examine the performance of various combinations of approaches on the test data. Full results are found in Table 2.

**Provided Train Data + Precision Hybrid Approach achieves highest F1.** Our highest achieved F1 score, on the test data, is 50.7. This is achieved via a BLSTM trained on the provided training data (Toxic Train) which uses the precision hybrid approach to find toxic spans. The next highest was from both the OLID BLSTM and Toxic Train BLSTM both using attention and achieving an F1 of 50.1.

Missed Spans
See a shrink you <i>*pathetic*</i> <troll>.
<Total> <i>*rubbish*</i> ! The birther bit was started by Crooked Hillary and perpetuated by Zero his own damn self.
and he is to <i>*stupid*</i> to know it. The guy is a total <poser> and a <i>*fool*</i>
Along with 'fat, dumb and happy'. Or maybe ' <i>*stupid*</i> ' is as <i>*stupid*</i> does'?
<Your an> <i>*idiot*</i> . The top 1% will get the biggest break while the rest of us struggle to pay rent.
In this case, anyway, I have bad news for you. <URStupid>
Sick <f**ks> like this should just be shot. What a waste of skin, air, and taxpayers money.
Falsely Detected Spans
Yes....it is to be expected....oddly enough it happens every time he opens his mouth and says <dumb> >thin<gs.
"""">Trump< is up to the task"" LMAO, you are an id <i>*iot*</i> >,"<
You are a <i>*simpleton*</i> , who is full of racial >hatred<. That is too bad. You make America a worse place.
I always knew you were a >dirty< cop.
By the way, Nazis were and are fascists so if you are one, your a <i>*fascist*</i> . >Scum< like this and Trumpler need to be run out of our country and given to ISIS to use as chew toys
McCain is a >pig<.
LNG is much better for the environment than diesel, yet these >stupid< hippies complain.

Table 3: Examples of missed spans and falsely detected spans. <Red> indicates missed tokens. *\*Blue\** indicate correctly identified spans. >Green< indicates wrongly asserted spans. Note that \*\*, <>, >< are not originally in text and are included for accessibility.

**OLID shows strongest transferability.** The OLID trained BLSTM achieved the highest F1 scores across all methods versus the other transfer sets, with an average of 46.8 (range: 44.7 - 50.1). The next highest set was Founta (avg: 39.7, range: 34.6 - 44.3). Surprisingly, Kaggle-Toxic (avg: 38.3, range: 22.5 - 44.6) arrives at third highest. This is surprising because the goal of the task is to identify “toxic” spans and Kaggle is toxic data, while OLID is offense, and Founta is abuse/hate data. The Gab and Davidson BLSTMs perform poorly in comparison only achieving average F1s of 19.0 and 18.8 respectively.

**Precision Hybrid Averages Highest F1 Score.** On average, the precision hybrid method outperforms other methods across datasets. The precision hybrid approach averages a F1 score of 35.4 (range: 19.0 - 49.8). The attention approach is second highest (avg: 35.8, range: 17.4 - 50.1), followed by the greedy remove (avg: 34.6, range: 18.6 - 44.7). This points to precision hybrid as a solid approach when the transfer strength is unknown.

**Indirect methods pale in comparison to more direct systems.** Although we examined various combinations of systems, our highest F1 (50.7) is still far less than the top submitted system (HITSZ-

HLT) 70.8. This difference is most likely due to our system aiming to find toxic spans indirectly, rather than directly training on the spans.

## 4 Error Analysis

To determine how to improve our system, we examine the spans commonly missed in the test set as well as spans commonly predicted wrongly. We examine errors in our top system (Toxic Train - Precision Hybrid).

### 4.1 Spans Missed

We find our system misses spans for several reasons. We present examples in Table 3.

#### Translation from tokens to character positions.

One reason for missed spans is the difficulty between translating toxic tokens to toxic spans. As our methodology finds specific tokens toxic, it needs to back-translate that to character positions in the original text. However, the text is pre-processed to help our system analyze it and therefore problems occur when lining up the characters.

**Toxic words are part of a phrase.** Another reason for missed spans is they are part of a toxic phrase and others part of the phrase are more toxic. For example, “Total rubbish” is marked as entirely toxic, however, our system marks only “rubbish”.



Spans Missed			
Translation	Part of Phrase	Overshadowed by more toxic words	Non Standard Words
75	8	25	2
Spans Falsely Detected			
Non-Toxic becoming Toxic	Translation	Unclear Why Not Toxic	Toxic word in not toxic usage
19	23	22	30

Table 4: Frequencies of errors on 100 sampled texts. Note that multiple errors can exist in one text. Also, not all sampled texts have one of the prominent errors.

Another example is our system marking “pathetic” correctly but not “troll” in “pathetic troll”. Our system misses that these non-toxic words (e.g. “total”) can become toxic when combined in a phrase.

**Certain words overshadowed by more toxic words.** Some of the tokens which remain (e.g. “poser”) seem to be overshadowed by a stronger toxic word. For example, “and he is to stupid to know it. The guy is a total poser and a fool”, is marked as having three toxic spans (stupid, poser, fool), however, our system labels “stupid” and “fool” as toxic spans but not “poser”. This is most likely because our system views “stupid/fool” as more toxic words and thus assigns higher attention scores than for “poser”.

**Toxic words are non standard words.** Since the BLSTM uses word embeddings, non standard words are missed by our system. For example, “In this case, anyway, I have bad news for you. URStupid”, “URStupid” is marked as toxic, but it is a non standard word.

## 4.2 Spans Falsely Detected

Table 3 also contains examples of wrongly predicted spans by our system. We’ve identified reasons for these errors as well.

**Non-toxic words become associated with toxicity in training set.** Depending on the training set, our model can learn non-toxic words as toxic if they occur more frequently with the toxic label compared to the non-toxic. An interesting example is “trump” being marked as a toxic span, which means the training data provides “trump” in enough toxic texts for our system to associate that with toxicity.

**Translation to character indexing.** Similar to missing gold spans, our system’s predicted spans are often shifted off of the correct token. For example, in the phrase “dumb things”, our system correctly identifies that “dumb” is a toxic span, but during translation back to character index, it

is shifted 4 characters to the right, so instead our system says “thin” (part of “things”) is the toxic span. This issue causes many similar misses.

**Toxicity is not always clear.** Another reason for wrongly identified spans is that the toxicity isn’t always clear. That is, words which appear to be toxic, are marked as non toxic in the gold labels. For example, in “I always new you were a dirty cop.”, calling someone a “dirty cop” is arguably toxic, and our system marks “dirty” as toxic, but the gold standard labels indicate there is no toxic span in the text. This could be attributed to toxicity being a subjective idea (Aroyo et al., 2019).

**Toxic words used in non-toxic ways.** A final prominent reason for falsely predicted spans is words that are sometimes toxic, appear in non toxic sentences. For example, in “... Gotta ask, you got back flow preventer on your plumbing or is that some of the stupid you can’t fix?”, our classifier tags “stupid” as toxic, but it is not. This category has an overlap as well with the previous unclear category since toxicity can be subjective. Where the annotators might label a word are being used in a non-toxic way, we might label it as toxic and therefore say it is unclear why the span is not counted correctly.

## 4.3 Quantifying Errors

To obtain an understanding of the frequency of the different types of errors we randomly sample 100 texts which contained missed spans and 100 texts which contained falsely predicted spans. The results can be found in Table 4.

For missed spans, we see the translation from tokens to character positions causing the largest problems, as it occurs in 75 out of 100 samples. Words overshadowed by more toxic words was the next highest error appearing in 25 of 100 samples.

For falsely detected spans, the errors are more evenly spread out. Toxic words used in non-toxic

manners caused the most false predictions (30 out of 100), followed by the translation from tokens to spans (23 out of 100). These errors highlight the importance of the translation step, and where our system could benefit the most for improvement in the future.

## 5 Leveraging Multiple Views via Ensemble of Datasets

Ensemble	F1
Train + OLID + Kaggle-Toxic	54.0
Train + OLID + Kaggle-Toxic + Founta	49.6
Train + OLID + Founta	53.6
All	39.3
Train + Kaggle-Toxic + Founta	50.7
OLID + Kaggle-Toxic + Founta	49.5
Train + OLID-Att + Kaggle-Toxic	<b>55.1</b>
Top Solo (Toxic Train - Prec. Hybrid)	50.7

Table 5: F1 scores of ensemble models. All systems use the Precision Hybrid method, except for OLID-Att which uses the attention method. Train = Toxic Train

As toxicity can vary, it follows that different toxic datasets can identify different aspects of toxic spans. We explore this idea further by creating a simple majority voting ensemble model. The ensemble model takes in the predictions of multiple models trained on different datasets, then indices are included if a simple majority votes for the index. A summary of the performance of these models is found in Table 5. Due to its performance, the Precision Hybrid method is used in all instances except OLID-Att which leverages the attention method.

**Ensemble outperforms top solo model.** The top ensemble combination achieves an F1 score of 55.1, which is higher than the top solo F1 of 50.7. This ensemble consists of top scoring solo datasets and their top scoring respective methods: Toxic train - Precision Hybrid, OLID - Attention, and Kaggle-Toxic - Precision Hybrid. This higher score helps demonstrate how the various toxic-based datasets can bring useful perspectives to the solution.

**Lower systems reduce performance of ensemble.** While an ensemble can increase performance, not all toxic-based datasets help the model. While the top three solo datasets achieve an F1 score of 54, adding Founta reduces it to 49.6. Adding the lowest performing sets (Gab and Davidson), reduces F1 to 39.3. Thus the datasets used in the ensemble require much consideration.

## 6 Related Work

We provide a brief look at work related to ours and highlight recent related ideas.

**Toxic Language:** In recent years toxic language classification has seen much focus. Organized tasks for offensive language classification were seen multiple years in a row in GermEval (Wiegand et al., 2018), OffensEval 2019 (Zampieri et al., 2019b), and OffensEval 2020 (Zampieri et al., 2020). Additionally, hate and abuse have been the focus of classification (Founta et al., 2018) (Davidson et al., 2017). For toxicity specifically, Karan and Šnajder (2019) examine detecting threads that could lead to toxic language in the future. van Aken et al. (2018) present specific challenges that remain for toxic classification.

**Methodology:** Our work draws from both robustness and text style transfer work. Robustness work looks at attacks of adversaries against classification (and other) systems. Hsieh et al. (2019) look at the robustness of attention based systems. Their “greedy-select” method for selecting words to change in obfuscation, is similar to our “greedy-remove” method. Style transfer often focuses on transferring text from one style to another. To change the style, some authors first remove or mask words which indicate the initial style from the text. Wu et al. (2019) accomplish this by leveraging both a frequency ratio method (same as ours) and attention. Though, the methodology is similar, we examine it in terms of toxic span identification where it has not been done before.

## 7 Conclusion

We leveraged various BLSTM methods to identify toxic spans. We find that a combination of a “Greedy-Remove” and “Attention” based methods with a focus on precision produces the highest overall results.

We examined how well various toxic related datasets are able to transfer to this task. We find that OLID, an offensive labeled dataset, transfers the highest, achieving an F1 only 0.6 lower than the provided dataset. Furthermore, we found that combining the viewpoints of the datasets in an ensemble model can increase the performance (F1 to 55.1).

Finally, we examined the errors our system made and the causes. We found that many errors were made when translating from tokens to character index. Future work would improve on this translation step to tighten the span indication.

## References

- Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. [Challenges for toxic comment classification: An in-depth error analysis](#). In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42, Brussels, Belgium. Association for Computational Linguistics.
- Lora Aroyo, Lucas Dixon, Nithum Thain, Olivia Redfield, and Rachel Rosen. 2019. [Crowdsourcing Subjective Tasks: The Case Study of Understanding Toxicity in Online Discussions](#), page 1100–1105. Association for Computing Machinery, New York, NY, USA.
- Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *ICWSM*.
- Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of twitter abusive behavior. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 12.
- Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.
- Mladen Karan and Jan Šnajder. 2019. [Preemptive toxic language detection in Wikipedia comments using thread-level context](#). In *Proceedings of the Third Workshop on Abusive Language Online*, pages 129–134, Florence, Italy. Association for Computational Linguistics.
- Brendan Kennedy, Mohammad Atari, Aida M Davani, Leigh Yeh, Ali Omrani, Yehsong Kim, Kris Coombs Jr, Shreya Havaldar, Gwenyth Portillo-Wightman, Elaine Gonzalez, et al. 2018. The gab hate corpus: A collection of 27k posts annotated for hate speech. *PsyArXiv*. July, 18.
- Joosung Lee. 2020. [Stable style transformer: Delete and generate approach with encoder-decoder for text style transfer](#).
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- D. Terdiman. 2018. [Here’s How Facebook Uses AI To Detect Many Kinds Of Bad Content](#).
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.
- Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. [Mask and infill: Applying masked language model for sentiment transfer](#). In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5271–5277. International Joint Conferences on Artificial Intelligence Organization.
- Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.
- Harish Yenala, Ashish Jhanwar, Manoj K Chinnakotla, and Jay Goyal. 2018. Deep learning for detecting inappropriate content in text. *International Journal of Data Science and Analytics*, 6(4):273–286.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffensEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 Task 12: Multilingual Offensive Language Identification in Social Media (OffensEval 2020). In *Proceedings of SemEval*.
- Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer.

# S-NLP at SemEval-2021 Task 5: An Analysis of Dual Networks for Sequence Tagging

Viet Anh Nguyen \*, Tam Minh Nguyen \*, Huy Quang Dao \*, Quang Huu Pham \*

AI Research Team

R&D Lab, Sun Asterisk Inc.

{nguyen.viet.anh-d, nguyen.minh.tam-b,  
dao.quang.huy-b, pham.huu.quang}  
@sun-asterisk.com

## Abstract

The SemEval 2021 task 5: Toxic Spans Detection is a task of identifying considered-toxic spans in text, which provides a valuable, automatic tool for moderating online contents. This paper represents the second-place method for the task, an ensemble of two approaches. While one approach relies on combining different embedding methods to extract diverse semantic and syntactic representations of words in context; the other utilizes extra data with a slightly customized Self-training, a semi-supervised learning technique, for sequence tagging problems. Both of our architectures take advantage of a strong language model, which was fine-tuned on a toxic classification task. Although experimental evidence indicates higher effectiveness of the first approach than the second one, combining them leads to our best results of 70.77 F1-score on the test dataset.

## 1 Introduction

Social Network sites are an integral part of our society. These platforms are often designed to maximize user interaction without sufficient means to moderate such interactions. The amount of users being cyber-bullied by toxic comments has reached an alarming proportion (Chan et al., 2021). To efficiently maintain the health of online communities, an automatic online-content filtering tool needs to be developed. Numerous previous attempts to resolve this issue have focused on toxic comment classification (Georgakopoulos et al., 2018; Chu et al., 2017; Pham et al., 2020; Risch and Krestel, 2020). Although these classification models are capable of detecting toxic comments, their outputs are not interpretable (Mathew et al., 2020).

On the other hand, Toxic Spans Detection (Pavlopoulos et al., 2021) is a task of locating toxic

segments in texts. With such a system, the moderators can easily highlight offensive words in comments, which is an essential and explainable assistance for automated comment rating. In this paper, we propose our two approaches to resolve the task. Our contributions are as follows:

- We investigate the effectiveness of our slightly customized Self-training (Wei et al., 2021) technique for a sequence tagging problem - Toxic Spans Detection.
- We explore the benefits of combining different word representations including Byte Pair Encoding (Sennrich et al., 2015), contextual character-level (Akbik et al., 2018), FastText (Bojanowski et al., 2016) and RoBERTa (Liu et al., 2019) word embeddings in order to utilize different syntactic and semantic information learned by these embedding methods.
- Taking advantage of a well-domain-adaptive pre-trained language model on a classification task (Unbiased-toxic-RoBERTa (Hanu and Unitary team, 2020)), we successfully integrate our two above-mentioned methods to achieve a high F1-score of 70.77 and rank 2nd at the Semeval 2021 Task 5: Toxic Spans Detection.
- Numerous exciting insights of the system's performance have been drawn with detailed error analysis.

## 2 Related Work

### 2.1 Word representation learning

Word2Vec (Mikolov et al., 2013) is among the earliest models for extracting continuous word representations. Although there have been numerous modern pre-trained text embeddings that outperformed Word2Vec in downstream tasks, it is still widely

---

\*equal contribution

used due to its simplicity and effectiveness (Akbik et al., 2018). However, Word2Vec fails to handle rare or out-of-vocabulary words. To address this problem, FastText (Bojanowski et al., 2016) learn a word representation as sum of its character n-grams embeddings. On the other hand, (Sennrich et al., 2015) utilizes Byte Pair Encoding, an alternative approach for learning sub-word representations.

Recent pre-trained language models learn context-sensitive word representations by utilizing different pretext tasks namely autoregressive language modeling (Radford et al., 2019; Akbik et al., 2018), masked language modeling (Devlin et al., 2018) on a large amount of unlabeled data. Those methods have led to significant improvements in a wide range of downstream tasks, including Text Classification (Howard and Ruder, 2018), Question Answering (Devlin et al., 2018) and Named Entity Recognition (Akbik et al., 2019b).

Unbiased Toxic RoBERTa (Hanu and Unitary team, 2020) is a language model that utilizes general pre-trained RoBERTa (Liu et al., 2019) to continually pre-train a toxic comment classification task on Civil Comments Dataset <sup>1</sup>. This toxic-domain-adaptive language model can be successfully employed to Toxic Spans Detection task whose domain is a subset of Civil Comments.

## 2.2 Self-training

Self-training, a semi-supervised method, incorporates the prediction of teacher models on extra available in-domain unlabeled data into the training of a student model (Wei et al., 2021). Self-training has been recently successfully applied in both Computer Vision and Natural Language Processing tasks, including Image Classification (He et al., 2018), Object Detection (Xie et al., 2020), Machine Translation (He et al., 2020), ect. Despite its merits, issues such as the lack of in-domain unlabeled data (Du et al., 2020) and unreliable-pseudo labels (Pham et al., 2021) are the main obstacles for the success of Self-training.

For sequence-tagging problems, there are various methods of coping with noisy-pseudo labels. Unlike classification tasks, noisy self-labeled data can be easily eliminated by removing those which have low confidence scores; there is a lack of a comprehensive means to determine this score for a sequence-labeling data point. In several recent re-

search, a deep reinforcement learning (Chen et al., 2018) and meta-learning (Wang et al., 2020) has been proposed to reduce “error propagation from noisy pseudo-labels” for sequence labeling tasks.

## 3 Methodology

In this section, we describe our proposed framework in detail. Firstly, we develop a simple but strong baseline to discover the effectiveness of different backbone models. Consequently, we build, extend, and customize our two methods on top of the best backbone and baseline model.

### 3.1 Baseline

We consider this task as a word-level binary classification problem even though the label annotation of the dataset is at character-level. Therefore, we first align character annotations to word annotations. We utilize a straightforward architecture, with a pre-trained language model as the backbone and a simple classifier on top of it. Specifically, let denote  $\mathbf{w} = \{w_1, w_2, \dots, w_m\}$  and  $\mathbf{y} = \{y_1, y_2, \dots, y_m\}$  with  $w_i, y_i$  is the word and its label at position  $i$  respectively, and  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  with  $x_j$  is the  $j^{th}$  subword tokens. Notice here that  $m$  and  $n$  can be different because language models learns subword representations instead of word-embeddings.  $\mathbf{h} = \{h_1, h_2, \dots, h_n\}$  is the set of contextual embedding for all tokens in  $\mathbf{x}$  (taken from the last layer’s output of the backbone) and  $\mathbf{p} = \{p_1, p_2, \dots, p_m\}$  with  $p_i$  is the set of all subword positions of the word at position  $i$ . To obtain a word-level embedding, we took the sum of its corresponding subword embeddings.

$$e_i = \sum_{j \in p_i} h_j$$

Then probability distribution of the word at position  $i$  is formulized as follow.

$$p(w_i) = \text{Softmax}(W_c \text{ReLU}(W_h e_i + b_h) + b_c)$$

With  $W_c, b_c$  and  $W_h, b_h$  are the learnable weights and bias respectively. We optimize the model by minimize the Cross Entropy loss between the ground-truth and model predictions.

### 3.2 Method 1: Feature-based Learning

We customize and extend the baseline model by constructing a standard Named Entity Recognition

<sup>1</sup><https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification>

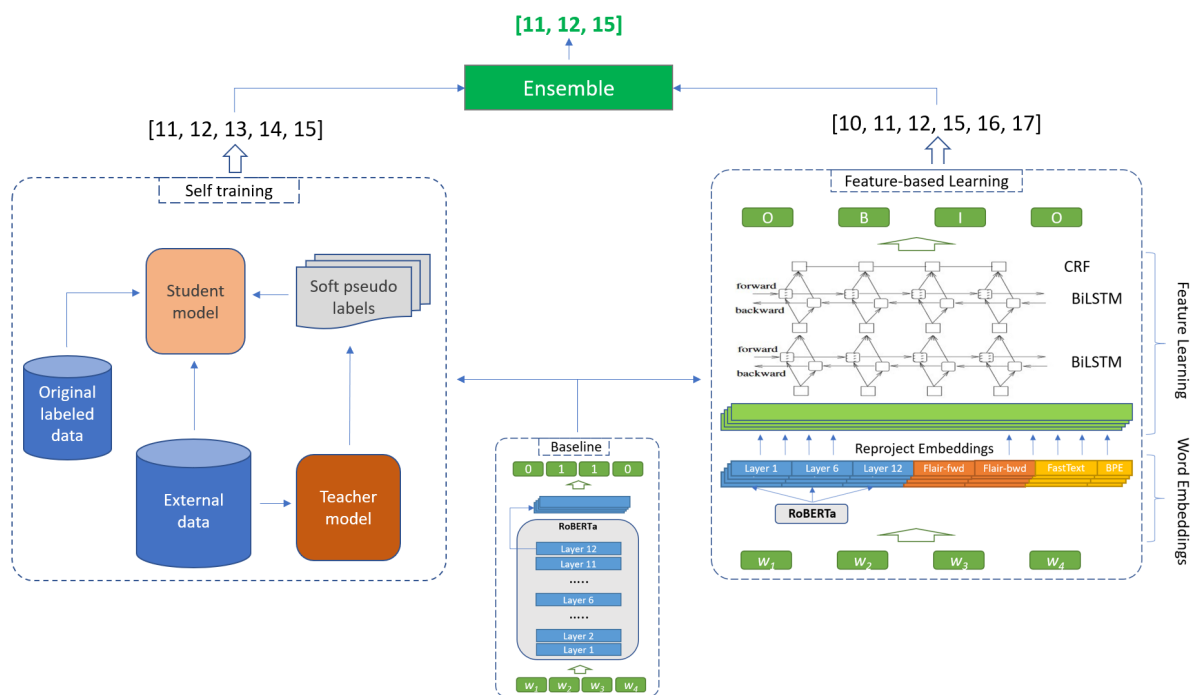


Figure 1: An illustration of our method. We start with a baseline, a simple sequence tagger utilizing Toxic RoBERTa as the backbone. In the Self-training branch, the teacher- the best-scored baseline, generates soft pseudo labels for the student to learn. On the other hand, the Feature-based Learning model concatenates the input vector with different embedding methods i.e. Flair, FastText and BPE, then trains the Named Entity Recognition task. Predicted character offsets (for each sentence) of two models are combined using Intersection Union (Ensemble Section) to obtain the final prediction.

model using Flair package (Akbik et al., 2019a)<sup>2</sup> in which each span is an entity encoded in IOB format. The model consists of two parts: input representation using diverse embeddings and a feature-based model.

### 3.2.1 Input Representation

In represent both syntactic and semantic information of a word, we combine embeddings extracted by different word embeddings methods. These representations strengthen advantages of each other while mutually easing their weaknesses.

These word embeddings and their usage in our works are as follows:

- **Flair:** Contextual Flair model works on character level. We fine-tune two models ‘news-forward’ and ‘news-backward’, on the Next Character Prediction task (Akbik et al., 2018), with 600K toxic texts from the Civil Comment Dataset to adapt them to toxic comment domain.
- **Toxic RoBERTa:** To utilize contextual embeddings from Toxic RoBERTa, besides fea-

tures derived from the last layer as our baseline, we concatenate two more layers: the first one (layer 1) and the middle one (layer 6). This choice allows the feature learning to understand three levels of context-specificity (Ethayarajh, 2019). The final word representation is obtained by taking the sum of its subword embeddings.

- **FastText with Byte Pair Embedding:** It has been practically proven that combining contextual embeddings with static embeddings improves the performance of many NLP downstream tasks (Peters et al., 2018). We discard subword part, take only word vector part of a FastText model (pre-trained on Common Crawl dataset) for word representation and utilize an external English Byte Pair Embedding for out-of-vocabulary functionality. This combination performs as well as the original FastText while effectively reduces memory usage.

All of the above embeddings are concatenated to form a long vector for each word, which is digested by a feature learning model.

<sup>2</sup><https://github.com/flairNLP/flair>

### 3.2.2 Feature Learning

The feature learning part is a sequence-to-sequence model that takes a sequence of word vectors and learns higher-level features and inferences tags. We use a linear layer to reproject the word embeddings onto a vector space with dimensions equal to the length of concatenated word embeddings. Two follow-up BiLSTM (Hochreiter and Schmidhuber, 1997) (Dyer et al., 2015) blocks are added to learn high-level semantic-syntactic dependencies of the sequence. Finally, a Conditional Random Fields (Sutton and McCallum, 2010) layer, placed on top of the BiLSTMs, makes tag prediction for each word.

### 3.3 Method 2: Self-training With In-domain Unlabeled Data

#### 3.3.1 In-domain data retrieval

In-domain unlabeled data is one of the determining factors for Self-training. The Toxic Spans Detection task’s labeled dataset is a subset of toxic-and-severe-toxic-labeled data in Civil Comment Dataset (Pavlopoulos et al., 2021). To retrieve additional data, we first selected posts classified as toxic by at least half of its toxicity annotators. After removing texts in both train and trial labeled datasets from the retrieved data, we randomly select a subset of 30,000 unseen texts for the task. The choice of extra datasets’ size is heuristic and limited due to low-computing resources.

#### 3.3.2 Data filtering and soft label

We slightly customized the pseudo-labels distillation process applied in classification tasks (He et al., 2018) for the sequence-tagging problem. Instead of evaluating and selecting each text in unlabeled data, we use the teacher model’s post-softmax class probabilities to evaluate and select each word in a context. Specifically, if each word’s confidence score is greater than a threshold, we keep the back-propagation process through that word; otherwise, we ignore it. Notice here that the probabilities mentioned above are also utilized as confidence scores and pseudo-labels for the student training.

#### 3.3.3 Combine generated-labeled data with original-labeled data

The student model is trained on a combination of original-labeled and synthetic-labeled datasets. It has the same architecture as the teacher model except for increases in dropout rates of dropout layers and the hidden size in the model’s head classifier.

We chose the best checkpoint of the baseline model as teacher model.

### 3.3.4 Post-processing

For each continuous toxic-predicted span, we eliminate any existing punctuation at both its beginning and end. Additionally, to partially prevent our model from predicting common toxic comments’ targets as toxic spans, we exclude any predicted span in our predefined list of targets (described in details in the Appendices section). This list is based on the identity-targets list of toxic comments in the Civil Comments Dataset.

### 3.4 Ensemble Learning

We combine our two approaches by taking intersection (Intersection Ensemble) or the union (Union Ensemble) of predicted character offsets generated by best model results, from each method, to obtain the final offsets for each sentence.

$$\mathbf{S}_I = \mathbf{S}_1 \cap \mathbf{S}_2 = \{x : x \in \mathbf{S}_1 \text{ and } x \in \mathbf{S}_2\}$$

$$\mathbf{S}_U = \mathbf{S}_1 \cup \mathbf{S}_2 = \{x : x \in \mathbf{S}_1 \text{ or } x \in \mathbf{S}_2\}$$

With  $\mathbf{S}_I$ ,  $\mathbf{S}_U$ ,  $\mathbf{S}_1, \mathbf{S}_2$  are the intersection, union Feature-based Learning and Self-training offset predictions for one sentence of the ensemble model respectively.

Figure 1 illustrates our composed framework: the two approaches, built and extent on top of the baseline, are combined for the final predictions.

## 4 Experiments

### 4.1 Dataset

The original dataset contains 7939 annotated samples for training and 2000 unlabeled samples for testing. We use a small trial dataset, given by the task organizer which consists of 690 labeled samples, as our development set. We train our models on the training set, use the development set to find the best hyper-parameters, and finally make our submission on the private test set.

### 4.2 Experiment setup

This section focused on the hyper-parameters configurations of our two methods and is mentioned in the Appendices section.

### 4.3 System Configuration

Our experiments are conducted on a computer with Intel Core i7 9700K Turbo 4.9GHz, 32GB of RAM, GPU GeForce GTX 2080Ti, and 1TB SSD hard disk.

#### 4.4 Evaluation Metric

The evaluation metric of our system is defined, by the task organizer (Pavlopoulos et al., 2021), as follow:

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|}$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|}$$

$$\text{if } S_G^t = \{\phi\} \Rightarrow F_1^t(A_i, G) = \begin{cases} 1 & \text{if } S_{A_i}^t = \{\phi\} \\ 0 & \text{otherwise} \end{cases}$$

$$F_1^T(A_i, G) = \frac{1}{n} \sum_{t=1}^n F_1^t(A_i, G)$$

With:

- $S_{A_i}^t$ : character offsets of toxic post  $t$ , output of system  $A_i$
- $G^t$ : ground truth character offsets of toxic post  $t$
- $F_1^t(A_i, G)$ :  $F_1$  score of system  $A_i$ , with respect to ground truth  $G^t$  of post  $t$
- $F_1^T(A_i, G)$ :  $F_1$  score of system  $A_i$  on dataset  $T$
- $|\cdot|$ : set cardinality

#### 4.5 Results

##### 4.5.1 Baseline result

Table 1 indicates the performances of our baseline model with two different backbones, RoBERTa (Liu et al., 2019) and Unbiased Toxic RoBERTa (which is referred as Toxic RoBERTa for the rest of the paper) (Han and Unitary team, 2020). The toxic domain-adaptive pre-trained language model outperforms general RoBERTa by a large margin (up to 0.68), which sheds light on the necessity of adapting universal representations to task-specific domains.

Backbone	Private test F1-score
RoBERTa	68.62
Toxic RoBERTa	<b>69.30</b>

Table 1: Performances of the baseline model with two different backbones.

##### 4.5.2 Feature-based Learning result

We froze Toxic RoBERTa backbone in all experiments of feature-based learning except the last one. This latest experiment compares the differences in model performances between tuning and not tuning Toxic RoBERTa.

Word Embeddings	Private test score
Toxic RoBERTa	69.89
FastText w/ BPE	67.89
Flair	67.92
Toxic RoBERTa + Flair	69.99
Toxic RoBERTa + FastText w/ BPE	69.95
Toxic RoBERTa + Flair + FastText w/ BPE	<b>70.26</b>
Toxic RoBERTa (fine-tuned) + Flair + FastText w/ BPE	67.37

Table 2: Results of different embedding combinations for method one

Table 2 shows the feature-based model’s performance with different word embeddings and the gap in F1-score between feature-based and fine-tuning models. Our findings are as follows:

Toxic RoBERTa was the best feature extractor since using it achieved a competitive F1-score of 69.89. On the other hand, using only Flair results in a slightly better performance than FastText with BPE (67.92 and 67.89 respectively).

Adding more features (learned by Flair or FastText with BPE embeddings) to ones learned by Toxic RoBERTa improved F1-score (69.99 and 69.95 respectively). Ultimately, combining all the word-representations obtained the highest score at 70.26.

Fine-tuning RoBERTa dramatically decreased the performance (up to 3-4).



### 4.5.3 Self-training result

Table 3 presents the performance result of the 2nd method. Our choice of the teacher was the best-performed baseline model with 69.30 F1-score. Post-processing enhanced this performance, resulted in 69.44 F1-score. Self-training only leads to a better student with an improvement of 0.1 compared to the post-processed teacher model. We suspect that this unimpressive increase is due to the teacher model’s confirmation bias and the unsolved issue of noisy-pseudo labels (Pham et al., 2021).

Backbone	Private test F1-score
Teacher w/o Post processing	69.30
Teacher w/ Post processing	69.44
Student	<b>69.54</b>

Table 3: Performances of the teacher model with and without post-processing and student model.

### 4.5.4 Ensemble learning result

Table 4 illustrates the effectiveness of our ensemble methods. Intersection Ensemble results in a significant improvements of our system prediction (0.51 and 1.23 compared to Feature-based Learning and Self-training respectively) while Union Ensemble leads to a substantial decrease of F1-score (-1.14 and -0.42 compared to method 1 and 2 respectively). This exciting finding indicates that Intersection Ensemble can rule out numerous falsely positive tokens of our two models whereas Union Ensemble worsen the performance by integrate these false positives.

Method	Private test F1-score
Feature-based Learning	70.26
Self-training	69.54
Union Ensemble	69.12
Intersection Ensemble	<b>70.77</b>

Table 4: Performance of ensemble models with different ensemble methods.

## 5 Error Analysis

Carefully analyzing errors made by our ensemble model on the test dataset has shed light on our sys-

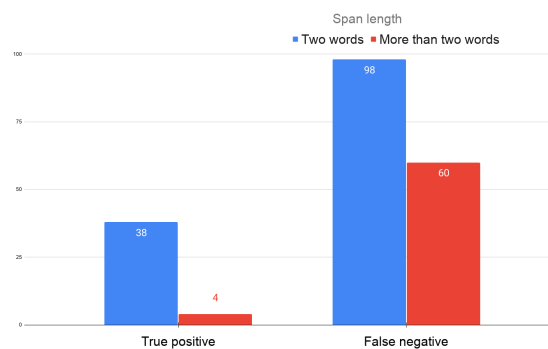


Figure 2: Analysis of the detection of lengthy toxic spans.

tem’s weaknesses. The following terms are defined to aid the reading of this section:

- Toxic span: ground-truth toxic span.
- Long span: contains at least two words.
- True positive span: all words in the span are correctly predicted as toxic.
- False negative span: at least one word in the toxic span are predicted as non-toxic.

### 5.1 Unsuccessful detection of lengthy toxic spans

Among false negative spans in our system’s predictions, 218 spans are short spans, which constitutes a small portion of the total short spans of the test set. In contrast, the other 158 spans are long spans while the total number of long spans of the test set is 200. This means our model had a bad performance on lengthy toxic spans with false negative rate up to 79%.

Figure 2 further investigates on long span failure. While 90% (38 spans/42 spans) of our correctly predicted long spans contains only two word, this numbers for false-negative spans decrease to 62% (98 spans/158 spans). Longer spans are harder to predict. Failing to detect spans with more than three words contributes approximately 38% to the model’s false-negative.

Table 5 demonstrates several instances of the failure.

### 5.2 False negatives due to post-processing

Although experimentally showed its effectiveness, post-processing after Self-training inevitably excluded continuous predicted spans such as “NIGGERS” and “MUSLIMS” which are potentially a part of ground-truth toxic spans.

Ground-truth	Prediction
Bunch of cowards	cowards
burn them at the stake	burn them
big, fat, orange buffoon	None
you're a fecking write-off	None

Table 5: Several instances of errors in detecting long-spans

I wish <u>this moron</u> would have been shot to death by the US soldier instead of the other way around. I hope that 10 million is given to the soldier's family who actually deserve it.
Our <u>ignorant</u> , clueless legislators are working hard to ensure that Alaskans are as <u>ignorant</u> and clueless as the legislators themselves are. <u>Stupid</u> Alaskans are the only way these <u>moronic</u> legislators will continue to get re-elected.
<u>Trash</u> , and just not the regular <u>bigoted</u> flatulence, this crap you write is evil trash

Note: Underlines are the prediction of our models and bold text are our manually annotated toxic-spans.

Table 6: Several examples of our model predictions on no-span texts, which may have been mis-annotated

### 5.3 Failure due to mis-annotated spans

We notice our model predicted false positive tokens in 469 toxic comments and most of them (308 comments) are humanly annotated with no toxic spans. In our opinion, many of these texts are mis-annotated, which potentially lower the precision of our system.

Table 6 presents several examples of this issue. The underlines are our model predictions, while the bold text spans are our opinion of what toxic annotations should be for the given text. All these texts contain no toxic spans, according to the dataset's annotators.

## 6 Conclusion

In this paper, we proposed a system to resolve the SemEval task 5: Toxic Spans Detection. Our method utilized a pre-trained language model in toxic-domain and successfully combined two approaches Self-training and Feature-based Learning to achieve a high F1-score of 70.77. Finally, we provided insights into failure of the system and the task's potential falsely-negative annotations issue with careful error analysis.

Despite our success on the leader board, in future research, we determine to improve our model as follow:

- Investigate a solution for the noisy-pseudo label issue to enhance the performance of the

Self-training method.

- Combine Self-training with Feature-based Learning to learn a more robust toxic-span detection model.

### Acknowledgment

This work is partially supported by *Sun-Asterisk Inc*. We would like to thank our colleagues at *Sun-Asterisk Inc* for their advice and expertise. Without their support, this experiment would not have been accomplished.

## References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019a. [FLAIR: An easy-to-use framework for state-of-the-art NLP](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019b. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, page 724–728.
- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.
- Tommy K.H. Chan, Christy M.K. Cheung, and Zach W.Y. Lee. 2021. [Cyberbullying on social networking sites: A literature review and future research directions](#). *Information Management*, 58(2):103411.
- Chenhua Chen, Yue Zhang, and Yuze Gao. 2018. [Learning how to self-learn: Enhancing self-training using neural reinforcement learning](#). pages 25–30.
- T. Chu, Kylie Jue, and Max L. Wang. 2017. Comment abuse classification with deep learning.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. [Self-training improves pre-training for natural language understanding](#).
- Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. [Transition-based dependency parsing with stack long short-term memory](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.
- Kawin Ethayarajh. 2019. [How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings](#).
- Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. [Convolutional neural networks for toxic comment classification](#). In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence, SETN '18*, New York, NY, USA. Association for Computing Machinery.
- Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.
- Junxian He, Jiatao Gu, Jiajun Shen, and Marc’Aurelio Ranzato. 2020. [Revisiting self-training for neural sequence generation](#).
- Kaiming He, Ross Girshick, and Piotr Dollár. 2018. [Rethinking imagenet pre-training](#).
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- Jeremy Howard and Sebastian Ruder. 2018. [Fine-tuned language models for text classification](#). *CoRR*, abs/1801.06146.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. [Hatexplain: A benchmark dataset for explainable hate speech detection](#).
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#).
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Hieu Pham, Zihang Dai, Qizhe Xie, Minh-Thang Luong, and Quoc V. Le. 2021. [Meta pseudo labels](#).
- Q. H. Pham, V. Anh Nguyen, L. B. Doan, N. N. Tran, and T. M. Thanh. 2020. [From universal language model to downstream task: Improving roberta-based vietnamese hate speech detection](#). In *2020 12th International Conference on Knowledge and Systems Engineering (KSE)*, pages 37–42.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Julian Risch and Ralf Krestel. 2020. Toxic comment detection in online discussions.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. [Neural machine translation of rare words with subword units](#).

Charles Sutton and Andrew McCallum. 2010. [An introduction to conditional random fields](#).

Yaqing Wang, Subhabrata Mukherjee, Haoda Chu, Yuancheng Tu, Ming Wu, Jing Gao, and Ahmed Hassan Awadallah. 2020. [Adaptive self-training for few-shot neural sequence labeling](#).

Colin Wei, Kendrick Shen, Yining Chen, and Tengyu Ma. 2021. [Theoretical analysis of self-training with deep networks on unlabeled data](#).

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. 2020. [Self-training with noisy student improves imagenet classification](#).

## A Appendices

To form our target list in 3.3.4, we include original form, plural form, upper or lower case of each word in the table 7 to that list.

Target Identities	Male, female, transgender, heterosexual, homosexual, gay, lesbian, bisexual, Christian, catholic, jewish, Muslim, Islam, hindu, buddhist, atheist, black, white, asian, latino, Nigger, Mexican.
-------------------	--

Table 7: Common target identities in Civil Comment dataset.

Table 8 describes hyperparameter configuration for training. For Feature-based Learning, Flair embeddings are fine-tuned before training the NER model. Turn into Self-training, the best baseline model is used as the teacher. If not specified, the corresponding hyper-parameter value is used for training both baseline and student models

Task	Hyperparameter	Value	Description
Fine-tune Flair	pre-trained weights	"news-forward"/ "news-backward"	Initial weights of the Flair models.
	sequence_length	250	Length of character sequences
	mini_batch_size	500	Size of batches during training
	learning_rate	20	Initial learning rate
	patience	10	Number of epochs without improvement
	max_epochs	5	Number of maximum training epochs
	optimizer	SGD	Optimizer used for training
	scheduler	AnnealOnPlateau	Learning rate scheduler
Train NER model	dropout	0.3995	Probability of an element to be zeroed
	locked_dropout	0.4413	Probability of entire parameters in embedding space to be zeroed
	word_dropout	0.0677	Probability of entire words (or characters) in embedding space to be zeroed
	learning_rate	0.0005	Initial learning rate
	min_learning_rate	1e-07	Minimum learning rate to terminate training
	mini_batch_size	32	Size of batches during training
	max_epochs	50	Number of maximum training epochs
	optimizer	AdamW	Optimizer used for training
scheduler	AnnealOnPlateau	Learning rate scheduler	
Baseline + Self-training	hidden_size_T	150	Size of the linear projection of the teacher's head classifier
	hidden_size_S	160	Size of the linear projection of the student's head classifier
	learning_rate	1e-05	The learning rate used for training
	optimizer	AdamW	Optimizer used for training
	scheduler	None	No learning rate scheduler used
	dropout_T	0.3	Dropout rate of all dropout layers in the teacher head classifier
	dropout_S	0.4	Dropout rate of all dropout layers in the student head classifier
	max_epochs	5	Total training epochs
	label_smoothing	0.15	Label smoothing coefficient
	confidence_threshold	0.7	Use in obtaining extra data for student model, all words with post-softmax score (calculated by the teacher model) less than this threshold will be ignored
batch_size	8	Size of batches during training	

897  
Table 8: Hyperparameters for feature-based model training.

# UAntwerp at SemEval-2021 Task 5: Spans are Spans, stacking a binary word level approach to toxic span detection

Ben Burtenshaw and Mike Kestemont

Antwerp Centre for Digital Humanities and Literary Criticism

University of Antwerp

Prinsstraat 13, 2000

Antwerp (Belgium)

firstname.lastname@uantwerpen.be

## Abstract

This paper describes the system developed by the Antwerp Centre for Digital humanities and literary Criticism [UAntwerp] for toxic span detection. We used a stacked generalisation ensemble of five component models, with two distinct interpretations of the task. Two models attempted to predict binary word toxicity based on ngram sequences, whilst 3 categorical span based models were trained to predict toxic token labels based on complete sequence tokens. The five models' predictions were ensemble within an LSTM model. As well as describing the system, we perform error analysis to explore model performance in relation to textual features. The system described in this paper scored **0.6755** and ranked **26<sup>th</sup>**.

## 1 Introduction

**SemEval 2021 Task 5:** Toxic Spans Detection was organised by John Pavlopoulos and colleagues, and described in detail in their task description paper (Pavlopoulos et al., 2021). Competing teams were asked to develop systems capable of detecting spans of toxic text. Predictions were evaluated using a pairwise F1-score of toxic character offset predictions, described in section 5.1.

**Initial analysis** of the development data revealed that toxic spans were varied in content and not limited to single words. Though most examples contained single toxic words or phrases, others contained longer spans and complete sentences. Figure 1 illustrates this phenomena. With this in mind, we sort a strategy that combined longer span based detection with binary word classification.

Table 1 reveals that toxic spans were on average  $\tilde{3}$  times longer in the development set, whilst stop words were  $\tilde{4}$  times more frequent. Figures 8 and 9 shows the frequency of these features in relation to model performance.

```
# 38 | ... while brutal scum hold sway in our institu-
      | tions.
# 36 | Why even bother to give this woman a time of day.
      | Bury her in the cotton field where she belong.
# 31 | other religions aren't carrying
      | out the violent and extremist
      | doctrines as Islamists are.
# 34 | I hope he and other car theves read this,
      | I'll shoot you if I catch you in my vehicles
      | stealing then call the police.
```

Figure 1: Example of possible annotation rationales.

		dev			test		
		mean	total	std	mean	total	std
TOX	words	3.22	25547	9.37	1.24	2488	3.59
	stop_ws	0.98	7802	3.82	0.26	529	1.73
NOT	words	32.52	258165	34.39	31.44	62884	30.87
	stop_ws	16.19	128533	17.55	15.45	30909	15.75
support		7939			2000		

Table 1: Macro values for toxic and non-toxic spans for development and test data. Mean average, total number, and standard deviation are shown for all words and stop words. Support is shown from development and test data as the total number of samples

**Strategy** We combined models that used antithetical contexts, i.e. full sequences, and shorter ngram sequences before and after a given word. This approach is based on the hypothesis that their predictions would have a low correlation, and in turn, they would create ideal ensemble components.

**Results** The system described in this paper scored **0.6755** and ranked **26<sup>th</sup>**. We discovered that model correlation did play a factor in the accuracy of an ensemble approach; however, much of this performance increase was lost in transition to test data, where correlation increased on the most frequent type of examples. In section 5.3 we analyse model performance and correlation in relation to textual features.

## 2 Background

Toxic span detection is a development of binary toxicity detection which has garnered recent attention, in the form of shared-tasks and datasets (Wulczyn et al., 2017; Zampieri et al., 2019).

**Features** Teams were supplied with development data consisting of 7939 text samples in varying lengths up to 1000 characters, and tested on 2000 text samples.

**Target** Span detection asks systems to detect which specific series of characters are toxic, irrespective of the text’s overall toxicity. Figure 2 illustrates the target value for SemEval 2021 Task 5. Unlike Named Entity Recognition, systems were not scored on their performance at negative, beginning, middle, or end token detection. This target definition led to a focus on positive optimisation, where false positives were of more importance than true negatives. In section 5.3 on error analysis we compare model scores using a binary word level representation of toxicity, that scores both positive and negative prediction.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20  
 S a i d a v i l l a g e i d i o t .

Figure 2: Illustration of toxic span character offsets.

## 3 System overview

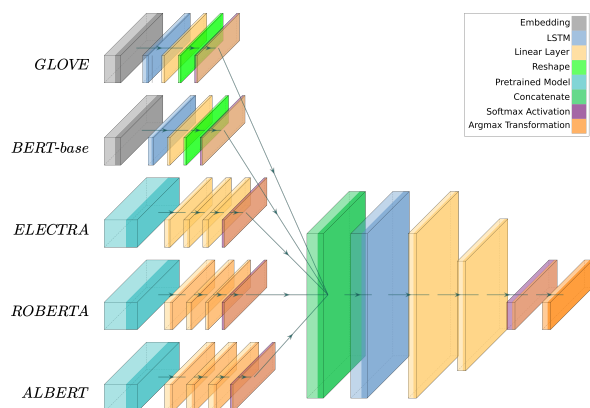


Figure 3: Model Diagram including all component models. Colours represent layer types and arrows represent training pipeline.

**Task Interpretations** We used two types of component models; binary word level models and categorical span based models, and combined those

in an LSTM network (Hochreiter and Schmidhuber, 1997). We used two word based models [GLOV, BERT] and three span based models [ALBE, ROBE, ELEC], the softmax output of all models were concatenated and supplied to an LSTM model [ENSE].

**Motivation** We intended for the word based models to learn local features in the tokens nearest the target word, and for the span based to learn the overall features that affected sub and multi word toxicity.

### 3.1 Baselines

To interpret the task we relied on the Spacy implemented baseline shared by the organizers and described in the task description paper (Pavlopoulos et al., 2021; Honnibal et al., 2020). The approach retrained the RoBERTa based `en_core_web_trf` model’s `ner`, `trf_wordpiecer`, and `trf_tok2vec` components, producing f1-scores of 0.5630 on the development data and 0.6305 on test data. To Interpret the problem further, we implemented two simple baselines.

**Lexical Lookup** Using a subset of samples from the development data, we created a toxic words list from all words within toxic spans, except for stop words<sup>1</sup>. On the test data, we then classified words as toxic if they appeared within the aforementioned toxic words list. We then converted word offsets into character offsets. This approach achieved an F1-score of 0.4161 on the test data.

**SVM** Using Term Frequency to Inverse Document Frequency we created two document vector representations of toxic and non-toxic spans. Using a Support Vector Machine, we predicted the probability that a word vector appeared within a toxic or non-toxic document (Salton and McGill, 1986; Wu et al.). We then used a binary threshold of 0.5 and class weights based on relative label frequency to predict whether a word was toxic. This approach achieved an F1-score of 0.5489 on the test data.

### 3.2 Component Models

#### 3.2.1 Span Prediction

Span prediction models used the complete sequence of words, up to a maximum length, to pre-

<sup>1</sup>The toxic words list was created from the first 5800 samples of the development data. We used Spacy tokenisation and English stop words list, and we removed space and character offsets from predictions.

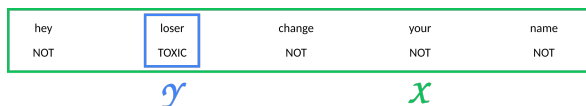


Figure 4: Illustration of toxic span prediction based on complete sequence.

dict toxic character offsets. Sequences were represented as token reference indexes, described in section 4.1. The target sequence was processed from character offsets into categorical arrays for toxic, non-toxic, and padding tokens. 4.1.

**Transformer Models** We selected three pre-trained transformer models (ALBERT, RoBERTa, ELECTRA) and fine-tuned them for this task with extra linear layers. We performed separate hyperparameter optimisation for each model, detailed in section 4.2. ALBERT is a lightweight implementation of a BERT model (Lan et al., 2020; Devlin et al., 2019) that uses feature reduction to reduce training time. ELECTRA is a further development of the BERT model that pre-trains as a discriminator rather than a generator (Clark et al., 2020). RoBERTa develops the BERT model approach for robustness, (Liu et al., 2019). During development we found that these three transformer models achieved the highest f1-scores in relation model correlation compared to alternatives. All models used the Adam optimizer (Kingma and Ba, 2017).

### 3.2.2 Binary Word Prediction

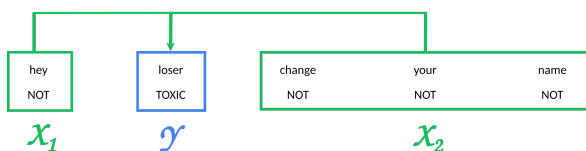


Figure 5: Illustration of toxic word prediction based on sequence before and after target word.

The binary word level models treated the task as word toxicity prediction based on a sequences of words before and after the target word. Figure 5 illustrates this approach. The target word toxicity was represented as a binary value. The sequence length before and after the target word was optimised for each model, and described in section 4.2.

### Siamese-LSTM with Glove Word Embeddings

A Siamese LSTM model used two networks based on separate glove embeddings of the sequence of

Task Representation	
input_text	"There are still morons"
target_labels	17, 18, 19, 20, 21, 22

BERT Representation					
input_tokens	there	are	still	mor	##ons
word_ids	0	1	2	3	3
target_labels	0	0	0	1	1

Figure 6: Input features and target labels for an example sequence, comparing a BERT specific token representation with the character offset representation defined by organisers (Pavlopoulos et al., 2021).

words before and after the target words (Bao et al., 2018; Baziotis et al., 2017).

**LSTM Finetuning BERT-base** An LSTM model was trained based on the output of a BERT-base model. The words before and after the target word were used as model features, and the target word toxicity was represented as a binary value (Devlin et al., 2019).

### 3.3 Ensemble Model

A Bidirectional LSTM model was used to predict token toxicity based on tokenised word features and component model predictions. The model used transformer style feature representations to predict a sequence of categorical representations for token toxicity, as described in section 4.1. The ensemble model relied on five fold cross validation, as described in section 4.2.

#### 3.3.1 Component model Predictions

Component model predictions were concatenated together as categorical representations of labels (not toxic, toxic, padding : 0,1,2). Each model's 3 dimensional output (number of samples, sequence length, number of labels) was permuted into a 4 dimensional matrix (number of samples, sequence length, number of labels, number of models).

## 4 Experimental setup

### 4.1 Pre-Processing

**Tokenisation** Text sequences were tokenised into character sequences using a BERT tokenizer and excess characters were replaced with a # character, as shown in Figure 6 (Devlin et al., 2019). Sequences were padded and truncated for uniformity to a length of 200 tokens. Longer sequences were handled separately, and predictions were combined in post-processing, described in section 4.4.



**Target Label Representation** To best suit the component models, we used a target representation based on the character sequences from the BERT tokenizer. Each word-like sequence was given a label based on its `word-id`, and converted into categorical binary arrays, or one-hot vectors. This is illustrated in Figure 6.

## 4.2 Training and Optimisation

**Cross Validation** We used stratified  $k$  fold validation of the development data to train all models. After optimisation, each component model’s predictions on the *test* portion of fold  $k$  were added to the *train* portion of the other folds. Producing unseen training features for the ensemble model. This process avoids overfitting in component models, and facilitates training an ensemble model on the complete development data (Fushiki, 2011; Pedregosa et al., 2011).

**Hyper-Parameter Optimisation** Model parameters were optimised for each fold of the development data and the best models were used by the ensemble model. Table 2 shows the optimum parameters for each model used on the test data. We used Bayesian optimization for each fold of the development data to find optimum parameters (Snoek et al.). Component models were selected based on their f1-score and prediction correlation to other models. The ensemble model was trained on the predictions of the optimum model for each fold of the development data, expanded on in Section 4.3.

method	span-based			word-based		
	ELEC	ROBE	ALBE	GLOV	BERT	ENSE
dropout	0.05	0.40	0.23	0.4	0.3	0.23
epochs	4	4	4	20	6	4
layers	2	2	3	3	3	3
nodes	9	3	6	20	3	6
neg_weight	1.00	0.92	1.12	0.6	1.0	1.0
pos_weight	1.00	1.24	0.94	6.0	1.0	1.0
dev_F1	0.665	0.663	0.682	0.647	0.656	0.702
test_F1	0.673	0.662	0.672	0.637	0.634	0.675

Table 2: Table of the best model parameters. Pairwise F1 scores are shown for all span based models.

## 4.3 Prediction

To predict spans for submission, a version of each component model optimised for each fold of the development data was supplied the test data and their outputs were averaged. The ensemble model

was then supplied component model predictions and tokenised text sequences.

## 4.4 Post-processing

Model output was converted from 2 dimensional token-level categorical arrays ( $n$  tokens,  $n$  labels) into character offsets. The character offsets of each positively labeled token was then added to a list, as illustrated in Figure 6. The predictions of sequences that had been truncated during pre-processing, were combined and duplicates were removed.

## 5 Results

Table 3 reveals that the ensemble model achieved a similar score on both development and test data, while the ALBERT, ELECTRA, and baseline models improved in testing. Crucially, the  $\approx 5\%$  increase in f1-score from component models to ensemble, that we see on the development data, was not transferred to the test data.

	dev			test		
	F1	P	R	F1	P	R
<b>ENSE</b>	0.6736	0.6664	0.7000	<b>0.6755</b>	0.6538	0.7182
<i>ALBE</i>	0.6284	0.6966	0.6677	0.6684	0.6695	0.6995
<i>ELEC</i>	0.6390	0.6975	0.6936	0.6668	0.6459	0.7296
<i>ROBE</i>	0.6418	0.7047	0.6908	0.6192	0.5771	0.7386
<i>BERT</i>	0.6568	0.6209	0.6260	0.5568	0.4209	0.5260
<i>GLOV</i>	0.6378	0.5850	0.5547	0.4378	0.4850	0.5547
BASE	0.5523	0.6247	0.5630	0.6305	0.5969	0.6548

Table 3: Scores on development and test data. The final submitted system predictions [ENSE] are shown in bold and component models are shown in italic.

## 5.1 Task Specific Evaluation Metrics

Systems are evaluated with an F1 score of character offsets (Pavlopoulos et al., 2021). In cases where predicted spans are empty, 1 is given when true spans are empty and 0 is given if there are any true spans.

## 5.2 Model Correlation

Figure 7 reveals that the ensemble and ALBERT models have a high correlation, a logical outcome of their shared base layers; whilst word based models [BERT, GLOV] have a low correlation, reflecting their diverse interpretations.

ENSE	1	0.9	0.86	0.83	0.8	0.7	0.75
ALBE	0.9	1	0.85	0.82	0.79	0.7	0.74
ELEC	0.86	0.85	1	0.83	0.76	0.65	0.74
ROBE	0.83	0.82	0.83	1	0.72	0.61	0.71
BERT	0.8	0.79	0.76	0.72	1	0.76	0.72
GLOV	0.7	0.7	0.65	0.61	0.76	1	0.65
BASE	0.75	0.74	0.74	0.71	0.72	0.65	1
	ENSE	ALBE	ELEC	ROBE	BERT	GLOV	BASE

Figure 7: Model Correlation calculated using a macro average f1-score

### 5.3 Error Analysis

We performed error analysis to interpret the hypothesis that there are multiple annotation rationales; single toxic words, and longer offensive sentences, illustrated in Figure 1.

**Toxic Span Length** Figure 8 reveals that the length of toxic spans had an impact on model performance. Models were less accurate at detecting longer spans on both development and test data. Furthermore, the impact of this effect on test data was decreased as there were fewer longer toxic spans.

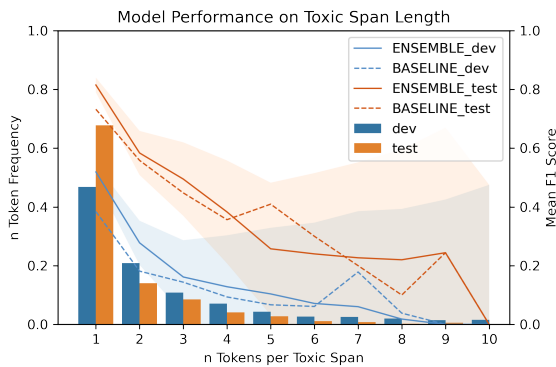


Figure 8: Model F1 score at  $n$  tokens per toxic span. Bars show the frequency of  $n$  tokens in development and test data. Shaded areas shows standard deviation of the f1-score for the ensemble model.

**Stop Words in Toxic Spans** The frequency of stop words in toxic spans also affected model performance. Figure 9 reveals that, where present, spans with more stop words caused lower model accuracy.

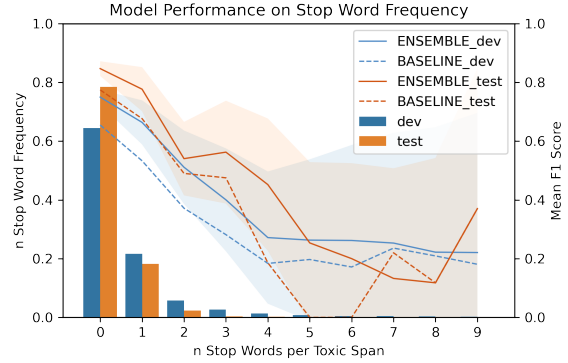


Figure 9: Model F1 score at  $n$  stop words per toxic span, and  $n$  stop word frequency.

**Binary Token Level Evaluation** By using token level scoring we are able to reveal how the models perform on both positive and negative tokens. Here, the target labels are represented as binary arrays; 1 for toxic tokens and 0 for non-toxic. We can not expect these calculations to align with character offsets, due to variance in tokenisation and parsing.

	NOT	TOX	NOT	TOX	
Baseline	0.93	0.55	0.97	0.63	Precision
	0.93	0.64	0.98	0.63	Recall
	0.97	0.57	0.97	0.69	f1-score
Ensemble	0.93	0.64	0.97	0.67	Precision
	0.93	0.71	0.98	0.68	Recall
	0.96	0.69	0.97	0.71	f1-score

Figure 10: Binary token level scores for precision, recall, and f1-score.

## 6 Conclusion

Our initial hypothesis, that combining word based and span based approaches would yield a significant performance boost, did not stand up. We measured a  $\tilde{5}\%$  increase in f1-score on development data, but this was not transferred to test data. In future work, we would look to a strategy that incorporated model transferability in component model selection, with the intention of better handling fluctuations in annotation rationale. Drawing on recent work (Fortuna et al., 2021).

## References

- W. Bao, W. Bao, J. Du, Y. Yang, and X. Zhao. 2018. [Attentive Siamese LSTM Network for Semantic Textual Similarity Measure](#). In [2018 International Conference on Asian Language Processing \(IALP\)](#), pages 312–317.
- Christos Baziotis, Nikos Pelekis, and Christos Doukouridis. 2017. [DataStories at SemEval-2017 Task 6: Siamese LSTM with Attention for Humorous Text Comparison](#). In [Proceedings of the 11th International Workshop on Semantic Evaluation \(SemEval-2017\)](#), pages 390–395, Vancouver, Canada. Association for Computational Linguistics.
- Kevin Clark, Minh-Thang Luong, and Quoc V Le. 2020. [ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS](#). page 18.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In [Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 \(Long and Short Papers\)](#), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Paula Fortuna, Juan Soler-Company, and Leo Wanner. 2021. [How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets?](#) [Information Processing & Management](#), 58(3):102524.
- Tadayoshi Fushiki. 2011. Estimation of prediction error by using K-fold cross-validation. [Statistics and Computing](#), 21(2):137–146.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. [Neural computation](#), 9(8):1735–1780.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength natural language processing in python](#).
- Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A Method for Stochastic Optimization](#). [arXiv:1412.6980 \[cs\]](#).
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A Lite BERT for Self-supervised Learning of Language Representations](#). [arXiv:1909.11942 \[cs\]](#).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A Robustly Optimized BERT Pretraining Approach](#). [arXiv:1907.11692 \[cs\]](#).
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In [Proceedings of the 15th International Workshop on Semantic Evaluation](#).
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. [Scikit-learn: Machine learning in python](#). [the Journal of machine Learning research](#), 12:2825–2830.
- Gerard Salton and Michael J McGill. 1986. [Introduction to modern information retrieval](#).
- Jasper Snoek, Hugo Larochelle, and Ryan P Adams. [Practical Bayesian Optimization of Machine Learning Algorithms](#). page 9.
- Ting-Fan Wu, Chih-Jen Lin, and Ruby C Weng. [Probability Estimates for Multi-class Classification by Pairwise Coupling](#). page 31.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex Machina: Personal Attacks Seen at Scale](#). [arXiv:1610.08914 \[cs\]](#).
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 Task 6: Identifying and Categorizing Offensive Language in Social Media \(OffensEval\)](#). In [arXiv:1903.08983 \[Cs\]](#).

# hub at SemEval-2021 Task 5: Toxic Span Detection Based on Word-Level Classification

Bo Huang, Yang Bai, Xiaobing Zhou\*

School of Information Science and Engineering

Yunnan University, Yunnan, P.R. China

\*Corresponding author: zhouxb@ynu.edu.com

## Abstract

This article introduces the system description of the hub team, which explains the related work and experimental results of our team's participation in SemEval 2021 Task 5: Toxic Spans Detection. The data for this shared task comes from some posts on the Internet. The task goal is to identify the toxic content contained in these text data. We need to find the span of the toxic text in the text data as accurately as possible. In the same post, the toxic text may be one paragraph or multiple paragraphs. Our team uses a classification scheme based on word-level to accomplish this task. The system we used to submit the results is ALBERT+BILSTM+CRF. The result evaluation index of the task submission is the F1 score, and the final score of the prediction result of the test set submitted by our team is 0.6640226029.

## 1 Introduction and Background

From the popularization of the Internet to the first year of the mobile Internet in 2011, the number of online social media and social media users has continued to grow. In the context of the ever-expanding user base, coupled with the free and interactive features of online social media communication. Therefore, online social media has exposed many issues worthy of our attention, such as the lack of communication standards and the out-of-control of information dissemination, which makes the dissemination of online social media prone to various negative functions (Baccarella et al., 2018).

The task of toxic span detection is to detect the span of text with toxic information in the text (Pavlopoulos et al., 2021). The goal of the task is to predict the beginning and ending character positions in the text as accurately as possible. Reviewing the content in online media can effectively avoid the spread of a series of negative information such as cyber violence, cyberbullying, and

false news. Audits are essential to promote healthy online discussions. However, the content and number of posts in social media are too large, and the manual review method obviously cannot achieve a good effect. Therefore, in combination with the development of modern technology, achieving a semi-automatic audit is the best solution.

## 2 Related Work

There are many different kinds of methods for identifying negative information in social media, but usually, these methods mainly focus on supervised learning. Simple SurfaceFeatures similar to the bag of words model can provide very clear and easy-to-understand information in text processing tasks. The general approach of this method is to merge multiple larger n-grams into a feature set (Nobata et al., 2016; Djuric et al., 2015). Use the artificial neural network method to train the word embeddings in the corpus. The purpose is to use the distance between vectors to indicate the semantic similarity of different words. Djuric et al. proposed a method of directly using embedding to represent the entire text and showed us the effect of this method (Sun et al., 2019).

Negative information in the text can be detected by the above methods. But more information needs to be obtained according to the context. The pre-trained language model based on the Transformer architecture has great advantages both at the word level and in context information (Wang et al., 2019). Therefore, in this task, we try to combine the pre-trained language model to complete the detection of toxic content.

## 3 Data and Methods

In this section, we introduce the data provided by the task organizer team to the participating teams, as well as the models and methods we use.

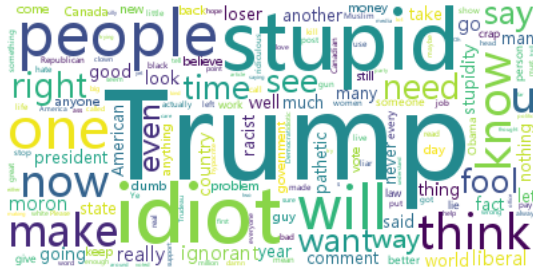


Figure 1: A word cloud diagram of the training set text data provided by the task organizer team. The result shown in the figure is the data after removing the stop words.

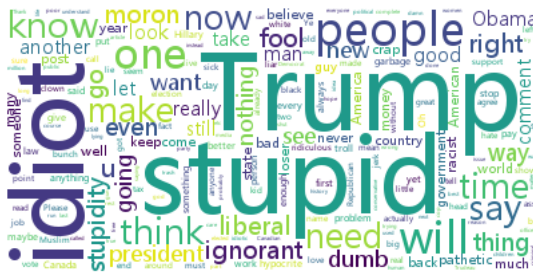


Figure 2: A word cloud diagram of the test set text data provided by the task organizer team. The result shown in the figure is the data after removing the stop words.

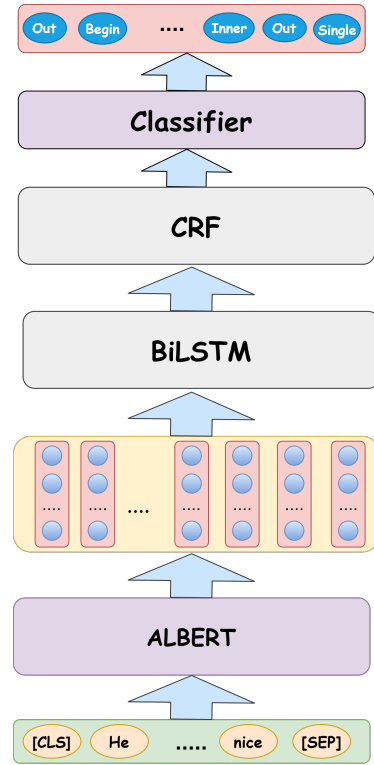


Figure 3: The model structure and data flow we used in the task.

### 3.1 Data Description

The task organizer team provides each team with training data sets and test data sets related to the “Toxic Spans Detection” task. The training data set consists of two parts, one is the text data of the post, and the other is the index position of the span of Toxic Spans. Each post corresponds to an index span data. There are some posts in the training set that do not contain toxic content, and there are one or more pieces of toxic content in the remaining posts. Also, in the index range of these toxic content, it may be a phrase, a sentence, or a word. The length of the post is not the same. Compared with the training data set, the test set only contains the text data of the posts. We need to use our method to predict the index span of the toxic content of posts in the test set. Table 1 shows the sample data of the data we used in the task.

There are 7939 and 2000 pieces of data in the training set and test set, respectively. We visualize the text data in the training set and the text data in the test set using word cloud graphs. The word cloud image clearly shows us the characteristics of word frequency distribution in the text data set. Regardless of the text data in the training set data or the text data in the test set data, some insulting

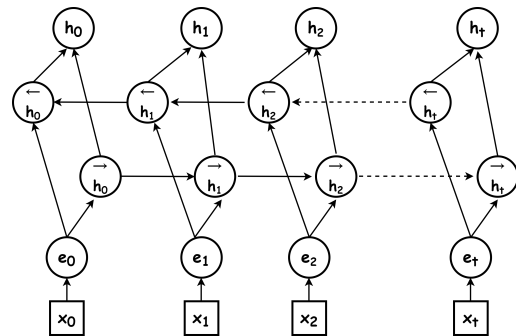


Figure 4: The BiLSTM structure and data flow

vocabulary, as well as some neutral vocabulary and human names (Trump) appear most. Those sentences with insulting words are usually detected as text with toxic spans. Some short sentences composed of words with neutral meanings and other phrases may also be recognized as text with toxic spans. The reason is because these sentences combined with some background information will convey unfriendly information. Figure 1 and Figure 2 show the word frequency information in the training set and the word frequency information in the test set.

Data category	Text	Spans
train	““““who do you think should do the <i>kill</i> ing?””””	[32, 33, 34, 35, 36, 37, 38]
train	“CROOKED Trump = GUILTY as hell. <i>pathetic</i> ”	[32, 33, 34, 35, 36, 37, 38, 39]
train	He is a scary maniac with a psychopath attitude.	[]
test	This <i>idiot</i> has no clue.	[5, 6, 7, 8, 9]

Table 1: Part of the data samples in the training set and test set provided by the task organizer team.

### 3.2 Methods

In our system, we use pre-processed data as the input to ALBERT. The architectures of ALBERT base and BERT base are both composed of 12-layer Transformer (Devlin et al., 2018; Lan et al., 2019). Compared with the BERT model, the result of the original embedding parameter  $P$  is the product of the vocabulary size  $V$  and the hidden layer size  $H$ . ALBERT factorizes the Embedding matrix by using a lower-dimensional embedding space of size  $E$  and then project it to the hidden space.

$$V * H = P \quad \rightarrow \quad V * E + E * H = P \quad (1)$$

Different from  $H=E$  in BERT, when  $H \gg E$ , the number of parameters of ALBERT has a significant reduction. Another big difference from BERT is that ALBERT’s default decision is to share all parameters across layers (Lan et al., 2019). Based on these improvements, the training effect of ALBERT is better than that of BERT. In terms of memory usage, the ALBERT pre-training model is also smaller than the BERT pre-training model.

Based on the structural characteristics of the RNN artificial neural network, the RNN network has great advantages in processing text data (Zaremba et al., 2014). But in the actual training process, a simple RNN network is difficult to converge. Because the loss value of the RNN network is continuously accumulated as the text sequence increases. Compared with the RNN network, LSTM artificial neural network has great advantages in model convergence and processing long text (Gers et al., 1999; Olah, 2015). LSTM is mainly composed of two key points, one is the cell state, the other is the gate unit. The information learned by the LSTM unit will be directly stored in the cell state, and we can input and update the value in the cell state. The gating unit plays a role in controlling how to update the value in the cell state. These gating units are composed of forget control gate, input control gate, and output control gate. The key

to the composition of the gating unit is the sigmoid function.

In our system, first, we use the preprocessed data as the input of ALBERT. Then, use the output of ALBERT as the input of BiLSTM. Next, the output of the BiLSTM model is used as the input of CRF. Finally, a classifier is used to classify the output results of CRF. The classifier needs to classify each word in the text into one of four different categories. These four categories are the beginning of the text span, the inside of the text span, the outside of the text span, and the toxic span formed by a single word. The architecture of BiLSTM, our model architecture and data flow can be seen in Figure 3 and Figure 4.

## 4 Experiment and Results

In this section, we will introduce the data preprocessing methods and experimental settings we used in the task and the final results.

### 4.1 Data Preprocessing

Because our model and method are to classify content at the word level. So we preprocessed the text data provided by the task organizer team. Preprocessing mainly involves dividing all words in each post into one of four categories (Begin, Out, Inner, Single). These four categories represent our specific description in Section 3.2, paragraph 4. Then split the processed training set into a new training set and a validation set. The split rule is to randomly extract part from the training set as the validation set, and the ratio of the training set to the validation set is 8: 2.

### 4.2 Experiment setting

We use preprocessed data as input to the model. During the training process, we adjust the parameters of the model according to the results of the model on the validation set. The learning rates used by the ALBERT-base, BiLSTM, CRF and classifier modules in the model are not the same. The learning rate used by ALBERT-base and BiL-

STM is  $3e-5$ , and the learning rate used by CRF and classifiers is  $1e-4$ . The maximum length of sentences input in the model is fixed at 120 words. The choice of this length comes from the length of the text in the data and the memory size of the GPU. Sentences that do not reach 120 words in length will be supplemented with zeros. Sentences longer than 120 words will be deleted. The epoch and batch during training are 10 and 32, respectively. The optimizer used in our experiment is Radam (Liu et al., 2019).

### 4.3 Results evaluation method

The evaluation index announced by the task organizer team is the F1 score. Let system  $A_i$  return a set  $S_{A_i}^t$  of character offsets, for parts of the post found to be toxic. Let  $G$  be the character offsets of the ground truth annotations. We compute the F1 score of system  $A_i$  with respect to the ground truth  $G$  for post  $t$  as follows, where  $|\cdot|$  denotes set cardinality.

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)} \quad (2)$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|} \quad (3)$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|} \quad (4)$$

If  $S_G^t$  is empty for some post  $t$  (no gold spans are given for  $t$ ), we set  $F1^t(A_i, G) = 1$  if  $S_{A_i}^t$  is also empty, and  $F1^t(A_i, G) = 0$  otherwise. We finally average  $F1^t(A_i, G)$  over all the posts  $t$  of an evaluation dataset  $T$  to obtain a single score for system  $A_i$  (Da San Martino et al., 2019).

### 4.4 Results

In the final results list announced by the task organizer team, a total of 91 team results are presented in the list. Our team’s F1 result score was 0.6640226029, ranking 37th. Table 2 presents the result score of our system on the validation set and the result score on the test set. Compared with the results of the top 3 teams in the ranking, our result is 0.0442802224 different from the optimal result. Table 3 shows the scores of the top three teams and our team on the test set.

Data	F1 score
Validation set	0.6821240031
Test set	0.6640226029

Table 2: The scores obtained by our system on the validation set and test set. The validation set comes from 20% of the training set provided by the task organizer team.

team	F1 score	Rank
HITSZ-HLT	0.7083028253	1
S-NLP	0.7077035474	2
hitmi&t	0.6984762534	3
hub(our method)	0.6640226029	37

Table 3: In the result list released by the task organizer team, the top 3 submitted test set prediction results scores and our submitted test set prediction results scores. A total of 91 participating teams submitted the prediction results of the test set.

## 5 Conclusion

This paper presents the system description submitted by our team to SemEval 2021 Task 5: Toxic Spans Detection. Our goal is to use our system to detect the span of toxic content as accurately as possible. We use a classification scheme based on word-level to complete the task. The system combines the pre-training language model (ALBERT) and BiLSTM+CRF commonly used in NLP tasks. The results we submitted proved the feasibility of our system, but compared with the optimal results, our method still has room for improvement. In future work, we will try to improve our methods to achieve better results.

## References

- Christian V Baccarella, Timm F Wagner, Jan H Kietzmann, and Ian P McCarthy. 2018. [Social media? it’s serious! understanding the dark side of social media.](#) *European Management Journal*, 36(4):431–438.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

- Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. [Hate speech detection with comment embeddings](#). In *Proceedings of the 24th international conference on world wide web*, pages 29–30.
- Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2019. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*.
- Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. [Abusive language detection in online user content](#). In *Proceedings of the 25th international conference on world wide web*, pages 145–153.
- Christopher Olah. 2015. Understanding lstm networks.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.
- Chenguang Wang, Mu Li, and Alexander J Smola. 2019. Language models with transformers. *arXiv preprint arXiv:1904.09408*.
- Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.



# Sefamerve ARGE at SemEval-2021 Task 5: Toxic Spans Detection Using Segmentation Based 1-D Convolutional Neural Network Model

Selman Delil\*

Birol Kuyumcu\*

Cüneyt Aksakalli\*

Sefamerve R&D Center  
Istanbul, Turkey

{selman.delil, birol.kuyumcu, cuneyt.aksakalli}@sefamerve.com

## Abstract

This paper describes our contribution to SemEval-2021 Task 5: Toxic Spans Detection. Our approach considers toxic spans detection as a segmentation problem. The system, Waw-unet, consists of a 1-D convolutional neural network adopted from U-Net architecture commonly applied for semantic segmentation. We customize existing architecture by adding a special network block considering for text segmentation, as an essential component of the model. We compared the model with two transformers-based systems RoBERTa and XLM-RoBERTa to see its performance against pre-trained language models. We obtained 0.6251 f1 score with Waw-unet while 0.6390 and 0.6601 with the compared models respectively.

## 1 Introduction

Unlike the text classification problems targeting to classify whole documents (Borkan et al., 2019; Schmidt and Wiegand, 2017; Pavlopoulos et al., 2019), toxic span detection is an NLP task focusing on capturing granular contents that make a text toxic. Proposed solutions may contribute to managing semi-automated moderations such as online discussions or news portals that are open to large participation and user comments. Therefore, the evaluation of systems that could accurately locate toxic spans within a text is considered a crucial step for this task (Pavlopoulos et al., 2021).

We adapt two solution approaches for the task. For the first approach, we consider toxic spans detection as a segmentation problem while in the second one we use transformers-based models. Our proposed model for the first approach uses character-based tokenized chunks as an input and outputs segmented text. The system uses a 1-dimensional (1-D) convolutional neural network adopted from U-Net architecture (Ronneberger

et al., 2015) commonly applied for semantic segmentation. We previously studied this approach, as we call Waw-unet, on text parsing problems for unstructured postal addresses, and achieved remarkable results (Delil et al., 2020).

In our second approach, we consider toxic spans as a single label Named-Entity Recognition problem. We employ several different transformers-based models and obtained better scores with RoBERTa (Liu et al., 2019) and XLM-RoBERTa (Conneau et al., 2020) network architectures. To see the difference between our main system and transformers-based models, we compared model achievements, and obtained 0.6251 f1 score with Waw-unet while 0.6390 and 0.6601 with the other models respectively. Following the final rankings, our best score ranked 44th among 91 submissions<sup>1</sup>.

## 2 Waw-unet Architecture

Waw-unet is a fully convolutional neural network architecture we designed by taking inspiration from U-Net architecture which was firstly developed for segmentation problems (Ronneberger et al., 2015). The U-Net network architecture is composed of two symmetric parts, which uses dimension reduction for the first half of the network, and then increases its dimension in the second half. In this architecture, the connections are taken from the convolutional layers on the encoding part, which also feeds each corresponding layer of the decoding part.

Similar to the pixel-based image segmentation, Waw-unet takes input samples, in our case text, and generates homogeneous masked regions for the targeted segment. However, unlike image processing which has multi-channel input, the network has 1-D input due to the single-dimensional nature of text

<sup>1</sup>Source code for our model is published on [https://github.com/birolkuyumcu/wawunet\\_for\\_toxicspan](https://github.com/birolkuyumcu/wawunet_for_toxicspan)

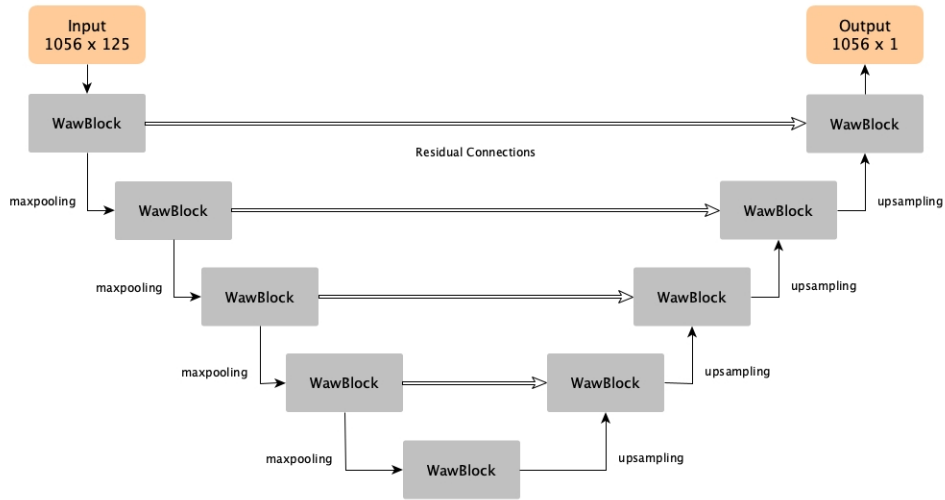


Figure 1: Waw-unet network architecture

data (Fig. 1). The outputs of the convolution layers are combined and passed through a 1-D convolution layer, and then we use batch normalization to accelerate training and prevent the objective function from getting stuck in local minima. After this stage, before the outputs send to the next block, if the output is in the encoder part their dimensions are reduced in half using max pooling, while doubled if it's in the decoder part by upsampling.

Although its architecture adapted from U-Net, we customize existing architecture by adding a special network block for text segmentation, as an essential component of the model. Waw-unet uses a special network block, which we call it Waw-block, to extract the attributes in the targeted text patterns. In our architecture, each waw-block contains three convolutional layers with different kernel sizes. Waw-unet learns input features through filter sizes of the 3, 5, and 7 as shown in Fig. 2.

## 2.1 Data Preparation

As we use a character-based system in our model, the total number of characters in the dataset and the maximum character length for each sample need to be determined. In the training dataset, the former was 1047, while the latter calculated as 125. Since our model has encoder-decoder architecture, to prevent matrix dimension problems, the input size has to be selected so that it can be divided by 2 until the end of the encoder part. Therefore, we defined max input size, the closest value as 1056, and the input matrix dimension as 1056 x 125. In accordance with the segmentation logic, the output character positions contain toxic spans masked as 1

and the other parts of the text masked as 0 (Fig. 3).

## 2.2 Model Training

The Tversky similarity index (TI) is used to calculate the loss function for training the network. It is an asymmetric similarity measure that is a generalization of Dice coefficient and Jaccard index (Tversky, 1977). To define Tversky loss function we use the following formulation:

TI : Tversky Index

TP : True Positive

FP : False Positive

$$TI = TP / (TP + a * FN + b * FP)$$

$$b = 1 - a$$

Here, we use  $1 - TverskyIndex$  as the loss function. The parameters  $a$  and  $b$  are used to provide weight to the represented classes. In our case, we determine  $a = 0.7$  to give weight to the false-negative classification so that the loss function is modified accordingly. Additionally, Dice similarity coefficient was used as a metric to judge the performance of the model training.

## 3 Transformers Models for NER

Toxic span detection can be adopted to NER problems by considering targeted toxic part of text as a predefined named-entity. We experiment with transformers models as an alternative for our model to see its performance. We use pre-trained models RoBERTa (Liu et al., 2019) and XLM-RoBERTa (Conneau et al., 2020) in our study utilising HuggingFace Trainer class.

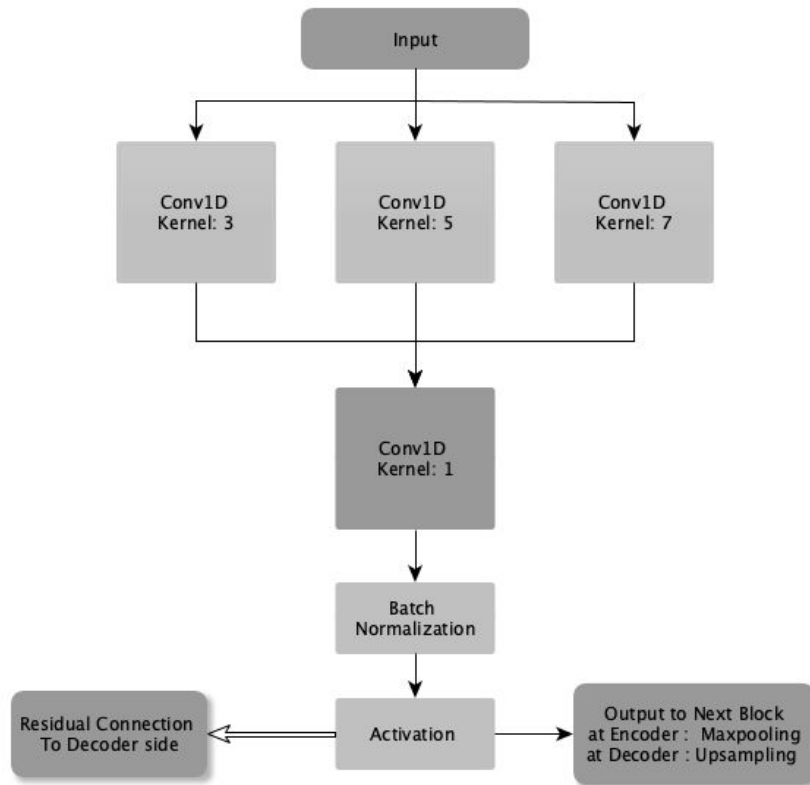


Figure 2: Waw-block architecture

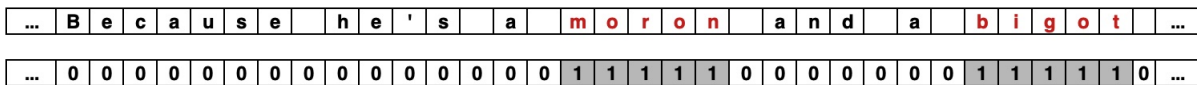


Figure 3: Waw-unet: Character-based masking

RoBERTa is an extension of pre-trained transformer model BERT with over more data and some configuration changes to the pre-training stages. The modifications include such as training longer and bigger batches, dynamically changing the masking pattern, and training on longer sequences (Liu et al., 2019). On the other hand, XLM-RoBERTa uses self-supervised training techniques designated to solve the cross-lingual understanding task. The model improves upon previous multilingual approaches by incorporating more training data and languages (Conneau et al., 2020).

To prepare toxic spans dataset for training, word labeling operation carried out by converting toxic spans into toxic words based on whether more than 50% of their characters labeled as toxic (Fig.4).

We use the Simple Transformers library (Rajapakse, 2019) to prepare our data for pre-trained models. Tokenized input containing the 3 columns—sentence\_id, words, and labels. Each value in words has a corresponding label value. In

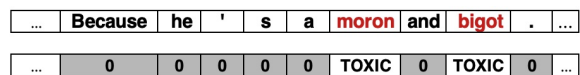


Figure 4: Transformers NER model: Word-based entity labeling

this data format, the sentence\_id determines which words belong to a given sentence (Fig. 5). We use the maximum sequence length of 128 for training and evaluation dataset.

## 4 Results

We evaluate our system as well as transformers models' performance on the SemEval-2021 Task 5: Toxic Span Detection trail dataset and also report the evaluation result on the blind test dataset. We use standard train/test split of the official release dataset of the Task for experiments.

For our main system model, Waw-unet, we start model training with 300 epochs and utilize early

sentence_id	words	labels
0	Another	0
0	violent	TOXIC
0	and	TOXIC
0	aggressive	TOXIC
0	immigrant	TOXIC
0	killing	0
0	a	0
0	innocent	0
0	and	0
0	intelligent	0
0	US	0
0	Citizen	0
0	.	0
0	.	0
0	.	0
0	.	0

Figure 5: Transformers NER model: Input data format

stopping and learning pause using Keras’s learning callbacks (Chollet et al., 2015). On the other hand, we trained transformers models with 8 batch size with 17 epochs. We gained the best score in 7th epoch on both pre-trained language models.

XLM-RoBERTa model gained best score 0.6601 while waw-unet and RoBERTa reached 0.6251 and 0.6390 respectively as shown in Table 1.

F1 For	Waw-Unet	RoBERTa	XLM-RoBERTa
Train	0.812	0.803	0.806
Trial	0.602	0.645	0.643
Test	0.625	0.639	0.660

Table 1: Model results

## 5 Conclusion

We framed the problem as a semantic segmentation task, and developed a unique approach to extract targeted spans from provided text data. Proposed system performs relatively well than expected against pre-trained transformers. Our models do not use any of the external dataset or automatic linguistic annotations, such as PoS or named entity tags. Overall, we showed that segmentation based systems can be used to address the toxic detection task. Our best submitted result was ranked 44th among 91 submissions, obtaining an average F1 score of 0.6601, 4.82 points behind the first ranked system.

For future studies, we’re planning to work on unsupervised training of the Waw-unet architecture on large datasets to compete with the pre-trained general language models.

## References

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced met-

rics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500.

François Chollet et al. 2015. Keras. <https://keras.io>.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#).

S. Delil, B. Kuyumcu, C. Aksakallı, and İ. S. Akçira. 2020. [Parsing address texts with deep learning method](#). In *2020 28th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

John Pavlopoulos, Nithum Thain, Lucas Dixon, and Ion Androutsopoulos. 2019. [ConvAI at SemEval-2019 task 6: Offensive language identification and categorization with perspective and BERT](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 571–576, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

T. C. Rajapakse. 2019. Simple transformers. <https://github.com/ThilinaRajapakse/simpletransformers>.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Anna Schmidt and Michael Wiegand. 2017. [A survey on hate speech detection using natural language processing](#). In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10, Valencia, Spain. Association for Computational Linguistics.

Amos Tversky. 1977. Features of similarity. *Psychological review*, 84(4):327.

# MIPT-NSU-UTMN at SemEval-2021 Task 5: Ensembling Learning with Pre-trained Language Models for Toxic Spans Detection

**Mikhail Kotyushev**  
Moscow Institute of  
Physics and Technology  
Moscow, Russia  
mkotyushev@gmail.com

**Anna Glazkova**  
University of  
Tyumen  
Tyumen, Russia  
a.v.glazkova@utmn.ru

**Dmitry Morozov**  
Novosibirsk State  
University  
Novosibirsk, Russia  
morozowdm@gmail.com

## Abstract

This paper describes our system for SemEval-2021 Task 5 on Toxic Spans Detection. We developed ensemble models using BERT-based neural architectures and post-processing to combine tokens into spans. We evaluated several pre-trained language models using various ensemble techniques for toxic span identification and achieved sizable improvements over our baseline fine-tuned BERT models. Finally, our system obtained a F1-score of 67.55% on test data.

## 1 Introduction

Toxic speech has become a rising issue for social media communities. Abusive content is very diverse and therefore offensive language and toxic speech detection is not a trivial issue. Besides, social media moderation of lengthy comments and posts is often a time-consuming process. In this regard, the task of detecting toxic spans in social media texts deserves close attention.

This work is based on the participation of our team, named MIPT-NSU-UTMN, in SemEval 2021 Task 5, “Toxic Spans Detection” (Pavlopoulos et al., 2021). Organizers of the shared task provided participants with the trial, train, and test sets of English social media comments annotated at the span level indicating the presence or absence of text toxicity. We formulated the task as a token classification problem and investigated several BERT-based models using two-step knowledge transfer. We found that preliminary fine-tuning of the model on data that is close to the target domain improves the quality of the token classification. The source code of our models is available at <https://github.com/morozowdmitry/semEval21>.

The paper is organized as follows. A brief review of related work is given in Section 2. The definition of the task has been summarized in Section 3. The

proposed methods and experimental settings have been elaborated in Section 4. Section 5 contains the results and error analysis respectively. Section 6 is a conclusion.

## 2 Related Work

Computational approaches to tackle text toxicity have recently gained a lot of interest due to the widespread use of social media. Since moderation is crucial to promoting healthy online discussions, research on toxicity detection has been attracting much attention. Our work is also related to hate speech and abusive language detection (Fortuna et al., 2020).

The toxic speech detection task is usually framed as a supervised learning problem. Moreover, fairly generic features, such as bag of words (Harris, 1954) or word embeddings (Mikolov et al., 2013), systematically yield reasonable classification performance (Fortuna and Nunes, 2018; Schmidt and Wiegand, 2017). To better understand the mechanisms of toxic speech detection, some scholars (Waseem et al., 2017; Lee et al., 2018; Karan and Šnajder, 2018; Swamy et al., 2019) compared different techniques for abusive language analysis. Neural architectures and deep learning methods achieved high results in this domain. Thus, Pavlopoulos et al. (2017a,b) explored the possibilities of deep learning and deep attention mechanisms for abusive comment moderation. Park and Fung (2017) proposed an approach to performing classification on abusive language based on convolutional neural networks (CNN). Chakrabarty et al. (2019) used Bidirectional Long-Short Term Memory network. Castelle (2018) experimented with CNN and Gated Recurrent Units. Some recent studies (Mozafari et al., 2019; Risch et al., 2019; Liu et al., 2019a; Nikolov and Radivchev, 2019) utilized pre-trained language models such as Bidirec-

tional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) to detect offensive or abusive language.

In recent years, the task of detecting and analyzing abusive, toxic, or offensive language has attracted the attention of more and more researchers. The shared tasks based on carefully curated resources, such as those organized at the SemEval (Zampieri et al., 2019; Basile et al., 2019), GermEval (Wiegand et al., 2018), EVALITA (Bosco et al., 2018), and OSACT (Mubarak et al., 2020) events, have significantly contributed to the progress of the field and to the enrichment of linguistic resources. In addition to the corpora collected for these shared tasks, Rosenthal et al. (2020) released a large-scale dataset for offensive language identification. Ibrohim and Budi (2018); Leite et al. (2020); Pitenis et al. (2020); Komalova et al. (2021) presented various datasets for abusive speech detection in non-English languages. Most of these datasets classify whole texts or documents, and do not identify the spans that make a text toxic.

### 3 Shared Task

The task focuses on the evaluation of systems that detect the spans that make a text toxic, when detecting such spans is possible. The goal of the task is to define a sequence of words (character offsets) that attribute to the toxicity of the text, for example:

- **Input.** “This is a stupid example, so thank you for nothing a!@#!@”.
- **Output.** [10,11,12,13,14,15,51,52,53,54,55, 56].

The sources of data were various posts (comments) from publicly available datasets. The provided dataset contains 10,629 posts split into training (7939), trial (690), and test (2000) subsets.

Inspired by Da San Martino et al. (2019), the organizers proposed to employ the F1-score for evaluating the responses of a system participating in the shared task. Let system  $A_i$  return a set  $S_{A_i}^t$  of character offsets, for parts of the post found to be toxic. Let  $G^t$  be the character offsets of the ground truth annotations of  $t$ . The F1-score of system  $A_i$  is calculated with respect to the ground truth  $G$  for post  $t$  as follows, where  $|\cdot|$  denotes set cardinality.

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)},$$

$$P^t(A_i, G) = \frac{S_{A_i}^t \cap S_G^t}{S_{A_i}^t},$$

$$R^t(A_i, G) = \frac{S_{A_i}^t \cap S_G^t}{S_G^t}.$$

The final F1-score is an average  $F_1^t(A_i, G)$  over all the posts  $t$  of an evaluation dataset  $T$  to obtain a single score for system  $A_i$ .

## 4 Methodology

The stated problem was modified from char-level to token-level binary-classification. The proposed solution utilizes a pre-trained language model with a classification head to classify tokens. Different configurations of BERT pre-trained as masked language models were considered as a backbone.

Due to the lack of available token-level labeled public datasets for toxic comment and the relatively small size and sparsity of dataset provided by the competition, the following training pipeline was proposed to enhance knowledge transfer. First, fine-tune pre-trained BERT on a larger-scale task of *toxic comment classification*, using the Jigsaw dataset<sup>1</sup> from which the competition data were constructed. Second, fine-tune obtained model to solve the actual *toxic tokens classification* problem. The exact training parameters are to be found below.

For the first step:

- remove texts occurred in spans dataset from classification dataset to prevent data leakage (so as spans dataset is sampled from classification dataset);
- 4 epochs, 200 tokens max length, 64 batch size, 10 gradient accumulation, mixed-precision FP16;
- default AdamW (Loshchilov and Hutter, 2017) with lr = 4e-5, Layer-wise Decreasing Layer Rate (Sun et al., 2019) with decay  $\eta = 0.95$  and cosine learning rate (LR) schedule with T = 4 epochs and constant LR after epoch 3;
- selected bert-base-uncased as best performance / speed ratio;
- the best model on validation selection each 0.1 epoch by AUC.

For the second step:

- hold-out  $\approx 14\%$  of data to train ensemble of models later;

<sup>1</sup><https://www.kaggle.com/c/jigsaw%2Dtoxic%2Dcomment%2Dclassification%2Dchallenge>

- out-of-5-fold training on the residual  $\approx 86\%$  of data;
- 4 epochs, 512 tokens max length, 16 batch size, 10 gradient accumulation, mixed precision FP16;
- default AdamW with  $lr = 4e-5$ , Layer-wise Decreasing Layer Rate with decay  $\eta = 0.95$  and cosine LR schedule with  $T = 4$  epochs;
- the best model on validation selection each 0.1 epoch by F1-score.

The final solution contains  $N \times K$  models, where  $N$  is the number of different backbone BERT architectures,  $K$  is the number of folds (5 in the current experiments). Obtained models are further to be ensembled using different strategies with validation on the single hold-out dataset:

- hard voting: final spans are selected as at least one model (spans union), as all the models (spans intersection) or as some intermediate methods with at least  $m$  models;
- soft voting: final probability is calculated as a weighted sum of models probabilities;
- train meta classifier.

## 5 Experiments and Results

Three pre-trained backbone BERT architectures were considered: bert-base-uncased, bert-large-uncased (Devlin et al., 2018), and bert-base pre-trained for Hate Speech Detection (dehate-bert) (Aluru et al., 2020). First step setup and results:

- select subset of Jigsaw toxic classification data: all the targets with toxicity score  $\geq 0.5$  ( $L = 135168$  objects) as class 1 and randomly sampled  $3 * L$  objects with toxicity score  $< 0.5$  as class 0;
- stratified 80% train, 20% validation;
- 0.968 AUC bert-base, 0.968 AUC bert-large, 0.942 AUC dehate-bert.

So as models except bert-base-uncased did not show compatible performance for token classification (and later for tests on the fold 0 did not show good F1-score for the actual task as well), later experiments were continued only for bert-base-uncased pre-trained model fine-tuned on token classification.

For step two results are following:

- train + trial, 8621 comments;
- average F1-score over 5 folds is 0.6714.

The experiments were conducted with Hugging-face transformers library (Wolf et al., 2019).

Many patterns in our results are expected, but some stand out. In general, our model is good at detecting obscene language and utterances that demean honor and dignity or denote low moral character. We noticed that our model is not very good at identifying the posts that have no toxic span annotations. According to the corpus description, in some toxic posts, the core message is conveyed may be inherently toxic. Thus, a sarcastic post can indirectly claim that people of a particular origin are inferior. Hence, it was difficult to attribute the toxicity of those posts to particular spans. In such cases, the corresponding posts were labeled as not containing toxic spans. Among our results, there are many examples where the model detected spans in not annotated posts, for example:

- “uhhh Hillary Clinton is a serial killer and thief”: [] (true annotation), [26, 27, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 44, 45, 46, 47, 48] (our annotation, “uhhh Hillary Clinton is a **serial killer** and **thief**”);
- “This goes way beyond just being an asshole skipper, dude must have some serious mental issues”: [] (true annotation), [35, 36, 37, 38, 39, 40, 41] (our annotation, “This goes way beyond just being an **asshole** skipper, dude must have some serious mental issues”).

In addition, some texts in the dataset raise questions of the annotation credibility, for example:

- “How the hell is this news? Am I supposed to be shocked that the Crown Prince of Bahrain or one of the world’s biggest celebrity superstars get’s better access to the State Department then I do? During which administration has this ever not been true? The media’s desperation to keep this election close is far past ridiculous” (training set, the toxic span annotation is underlined);
- “Yup. NVN used the Press. The Press was USED. Used like their sister on prom night! Idiots. All faux-erudite, not realizing they were being played” (training set, the original annotation is underlined);

Rank	Team	F1-score
1	HITSZ-HLT	0.7083
26	UAntwerp	0.67552
<b>27</b>	<b>MIPT-NSU-UTMN</b>	<b>0.67551</b>
28	NLRG	0.67532
	Avg result	0.57805

Table 1: Results on the test set.

- “And you are a complete **moron** who obviously doesn’t know the meaning of the word **narcissist**. By the way your bias is showing” (test set, the original annotation is underlined, the annotation of our model is highlighted in bold).

The final result of our model is presented in Table 1. As can be seen from the table, the systems of the participants produce close results. Our system achieved 67.55% of F1-score on the test set of this shared task that attracted 91 submitted teams in total. This value exceeded the average result by almost 10%.

## 6 Conclusion

This paper introduces our BERT-based model for toxic spans detection. As expected, pre-training of the BERT model using an additional domain-specific dataset improves further toxic spans detection performance. Experimenting with different fine-tuning approaches has shown that our BERT-based model benefits from the two-step knowledge transfer technique. An ensemble with spans intersection obtained our best result on the test data.

In our future work, we will evaluate various language models, such as distilled versions of BERT (Sanh et al., 2019; Jiao et al., 2020) and RoBERTa (Liu et al., 2019b).

## References

Sai Saket Aluru, Binny Mathew, Punyajoy Saha, and Animesh Mukherjee. 2020. Deep learning models for multilingual hate speech detection. *arXiv preprint arXiv:2004.06465*.

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Nozza Debora, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, Manuela Sanguinetti, et al. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.

Cristina Bosco, Dell’Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.

Michael Castelle. 2018. The linguistic ideologies of deep abusive language classification. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 160–170.

Tuhin Chakrabarty, Kilol Gupta, and Smaranda Muresan. 2019. Pay “attention” to your context when classifying abusive language. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 70–79.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.

Paula Fortuna, Juan Soler, and Leo Wanner. 2020. Toxic, hateful, offensive or abusive? what are we really classifying? an empirical analysis of hate speech datasets. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6786–6794.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Muhammad Okky Ibrohim and Indra Budi. 2018. A dataset and preliminaries study for abusive language detection in indonesian social media. *Procedia Computer Science*, 135:222–229.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4163–4174.

Mladen Karan and Jan Šnajder. 2018. Cross-domain detection of abusive language online. In *Proceedings of the 2nd workshop on abusive language online (ALW2)*, pages 132–137.

Liliya Komalova, Tatiana Goloshchapova, Leonid Motovskikh, Rostislav Epifanov, Dmitry Morozov, and Anna Glazkova. 2021. Mca workshop - toxic comments. Mendeley Data, V1.



- Younghun Lee, Seunghyun Yoon, and Kyomin Jung. 2018. Comparative studies of detecting abusive language on twitter. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 101–106.
- JA Leite, DF Silva, K Bontcheva, and C Scarton. 2020. Toxic language detection in social media for brazilian portuguese: new dataset and multilingual analysis. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 914–924. Association for Computational Linguistics (ACL).
- Ping Liu, Wen Li, and Liang Zou. 2019a. Nuli at semeval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 87–91.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Marzieh Mozafari, Reza Farahbakhsh, and Noel Crespi. 2019. A bert-based transfer learning approach for hate speech detection in online social media. In *International Conference on Complex Networks and Their Applications*, pages 928–940. Springer.
- Hamdy Mubarak, Kareem Darwish, Walid Magdy, Tamer Elsayed, and Hend Al-Khalifa. 2020. Overview of osact4 arabic offensive language detection shared task. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 48–52.
- Alex Nikolov and Victor Radivchev. 2019. Nikolov-radivchev at semeval-2019 task 6: Offensive tweet classification with bert and ensembles. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 691–695.
- Ji Ho Park and Pascale Fung. 2017. One-step and two-step classification for abusive language detection on twitter. In *Proceedings of the First Workshop on Abusive Language Online*, pages 41–45.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017a. Deep learning for user comment moderation. In *Proceedings of the First Workshop on Abusive Language Online*, pages 25–35.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017b. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 1125–1135.
- Zesis Pitenis, Marcos Zampieri, and Tharindu Ranasinghe. 2020. Offensive language identification in greek. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5113–5119.
- Julian Risch, Anke Stoll, Marc Ziegele, and Ralf Kretzel. 2019. hpidedis at germeval 2019: Offensive language identification using a german bert model. In *KONVENS*.
- Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Marcos Zampieri, and Preslav Nakov. 2020. A large-scale semi-supervised dataset for offensive language identification. *arXiv preprint arXiv:2004.14454*.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*, pages 1–10.
- Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583.
- Steve Durairaj Swamy, Anupam Jamatia, and Björn Gambäck. 2019. Studying generalisability across abusive language detection datasets. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 940–950.
- Zeerak Waseem, Thomas Davidson, Dana Warmsley, and Ingmar Weber. 2017. Understanding abuse: A typology of abusive language detection subtasks. In *Proceedings of the First Workshop on Abusive Language Online*, pages 78–84.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

# UIT-E10dot3 at SemEval-2021 Task 5: Toxic Spans Detection with Named Entity Recognition and Question-Answering Approaches

Phu Gia Hoang<sup>1,2</sup>, Luan Thanh Nguyen<sup>1,2</sup>, and Kiet Nguyen<sup>1,2</sup>

<sup>1</sup>University of Information Technology, Ho Chi Minh City, Vietnam

<sup>2</sup>Vietnam National University Ho Chi Minh City, Vietnam

{19520215, 17520721}@gm.uit.edu.vn, kietnv@uit.edu.vn

## Abstract

The increment of toxic comments on online space is causing tremendous effects on other vulnerable users. For this reason, considerable efforts are made to deal with this, and SemEval-2021 Task 5: Toxic Spans Detection is one of those. This task asks competitors to extract spans that have toxicity from the given texts, and we have done several analyses to understand its structure before doing experiments. We solve this task by two approaches, Named Entity Recognition with spaCy's library and Question-Answering with RoBERTa combining with ToxicBERT, and the former gains the highest F1-score of **66.99%**.

## 1 Introduction

The world of social media is overgrowing, and users easily express their opinions or feelings toward topics that they are concerned about. However, because of the freedom of speech, lots of toxic comments or contents are uncontrollably increasing. There are several kinds of research about the effect of toxic speech on users' health. In 2017, research about the impact of toxic language on health was conducted (Mohan et al., 2017). Sometimes, with toxic words, conversations can become cyberbullying, cyber threats, or online harassment, which are harmful to users. To reduce those negative impacts, there are abundant researches for classifying contents into toxic or non-toxic, and then they hide the whole text if it is toxic. However, that action may inhibit the freedom of speech. As a result, censoring only toxic spans is the better solution for this problem. Therefore, in SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021) we try to realize it.

About toxic contents on the internet, researches were only about binary toxicity classification. Still,

in task 5 of SemEval-2021, which is about toxic spans detection, we conduct more in-depth research into the toxicity, find exactly which parts of the text are toxic. As the NER approach and Question-Answering (QA) approach, we propose two approaches for solving this problem. We use RoBERTa (Liu et al., 2019) combining with ToxicBERT (Hanu and Unitary team, 2020), transfer learning models, for QA approach and spaCy's library (Honnibal and Montani, 2017) for NER approach.

We organize the paper as follows. Section 2 is related works that we consult for building the systems. The dataset and analyses are defined in Section 3. In section 4, we introduce our two proposed systems for toxic spans detection. Section 5 describes the results of the studies and analyses. Finally, in Section 6, we bring our work to a close.

## 2 Related Works

Researchers around the world these days have started to concentrate on toxic speech. It inflicts individual and group harm, damaging our social fabric (Tirrell, 2018). Several datasets for classifying toxicity on toxic speech on online forums, such as the dataset provided by Waseem and Hovy (2016) for English, BEEP! dataset for Korean by Moon et al. (2020), the dataset for Russian provided by Smetanin (2020), ToID-Br dataset for Brazilian Portuguese by Leite et al. (2020), and UIT-VICTSD, a dataset about constructive and toxic speech detection for Vietnamese (Nguyen et al., 2021).

Besides, there are shared tasks about toxic speech as well as hate speech such as these from SemEval, includes SemEval-2019 Task 5 Multilingual Detection of Hate (Basile et al., 2019), SemEval-2019 Task 6 Identifying and Catego-

ricing Offensive Language in Social Media (Of-fensEval) (Zampieri et al., 2019), SemEval-2020 Task 12 Multilingual Offensive Language Identification in Social Media (Zampieri et al., 2020), and SemeEval-2021 Task 5 Toxic Spans Detection (Pavlopoulos et al., 2021), which is the current task we have to deal with in this paper.

### 3 Dataset

The origin of this SemEval-2021 Task 5 dataset comes from the publicly available Civil Comments dataset (Borkan et al., 2019), which consists of 1.2M posts and comments. The data in this public dataset have no annotation of any toxic spans in toxic posts but do have post-level toxicity annotations, which mean showing which posts or entire of them are toxic. And the holders of this task retain 30K of them, which were annotated to be toxic or severely toxic by at least half of the crowd-raters from annotations of Borkan et al.

The task holders then randomly keep 10K posts from the 30K posts for annotating toxic spans. They employ three experienced crowd-raters per post from a third-party crowd-annotation platform, and they warn them about adult content. However, task organizers also claim that not all toxic posts are annotated with toxic spans.

The task for crowd-raters is to highlight toxic sequences of the comments, and if the comment is not toxic or should annotate the whole of it, crowd-raters have to check the appropriate box without highlighting any spans. Consequently, we have two columns, the spans column and the text column. The spans column has lists of numbers or null that reference toxic character offsets in the text column, and some of the given data are shown in the following table.

spans	text
[7, 8, 9, 10, 11, 12]	Pretty <b>damned</b> eloquent ... :)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28]	<b>He might fire you to the moon</b> , but you already have a head full of cheese!
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 98, 99, 100, 101, 102, 103, 104, 105, 106]	<b>Nauseating</b> and <b>disgusting</b> . Thank goodness the First Amendment permits people to demonstrate their <b>stupidity</b> .
[]	Not if they shoot you first...

Table 1: Examples for the given data.

The competitors receive two separate training

and test sets from organizers. In the training set, there are 7,939 records, and in the test set, there are 2,000 records. Furthermore, as mentioned in the data annotating process, one text that possibly has multiple toxic spans is highlighted. Figure 1 and Figure 2 illustrate the distribution of spans in the training and the test sets.

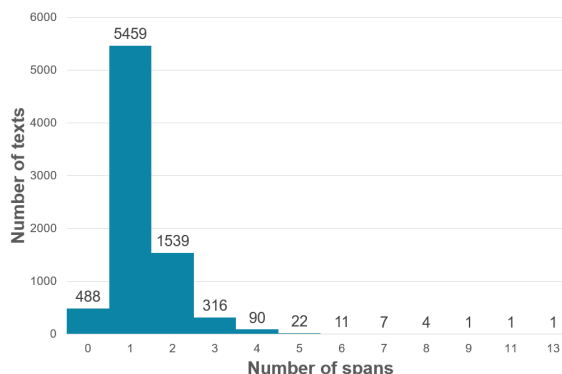


Figure 1: Distribution of spans in the training set.

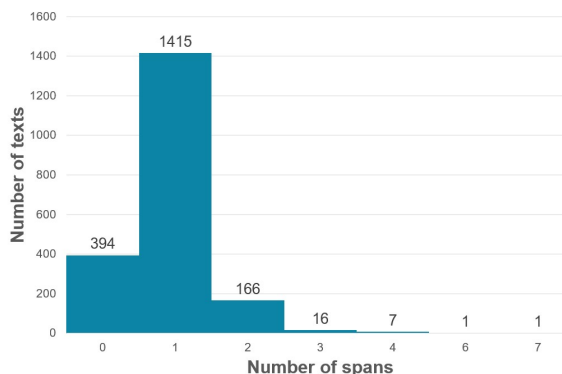


Figure 2: Distribution of spans in the test set.

For more details, according to Figure 1 and Figure 2, there is a significant number of single spans in each post, and it accounts for nearly 68.8% and 70.8% in the training set and the test set, respectively. It is also interesting to notice that the number of zero spans is not tiny, and the proportion of it in the training set is less than in the test set, more specifically, 19.7% in the test set and 6.15% in the training set.

Moreover, we also calculated the Jaccard score of text and spans in the given dataset for more in-depth analysis. The Jaccard score, also known as the Jaccard index or Jaccard similarity coefficient, was developed by Paul Jaccard (Jaccard, 1912) and it is a statistic used for measuring the similarity and diversity of sample sets as follows.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (1)$$

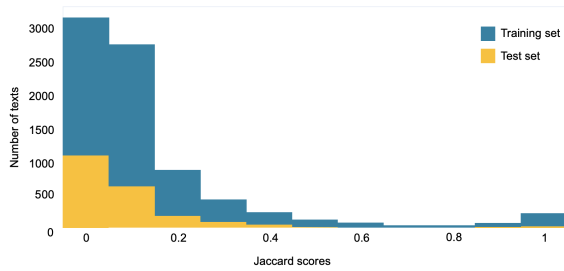


Figure 3: Histogram of Jaccard score of each record in the training set and the test set.

The histogram in Figure 3 illustrates that most of the data points have Jaccard scores in the range of 0 to 0.35, and the peak is at 0 to 0.05, which means toxic character offsets are just a fraction in each post even there are records annotated all characters of the post are toxic. There are 16 records in the test set and 212 records in the training set with Jaccard scores at 0.95 to 1.0. For that reason, just the toxic part(s) of the comments needs to be censored rather than the whole comment as in the traditional method.

## 4 Systems

In this paper, we propose two systems for the toxic spans detection task with NER and QA approaches. The first system is the QA approach based on RoBERTa and the second system is the NER approach based on spaCy’s library.

### 4.1 Question-Answering Approach Based on RoBERTa

With the QA approach, we use RoBERTa combining with ToxicBERT as the basis for the system. RoBERTa (Liu et al., 2019) is a transfer learning model and it is a replication study of BERT (Devlin et al., 2019). Unlike BERT, to improve the training performance, RoBERTa eliminates the Next Sentence Prediction (NSP) task of the pre-trained model BERT. ToxicBERT (Hanu and Unitary team, 2020) is also a transfer learning model, and it uses BERT as the main model for classifying toxicity. ToxicBERT has an outstanding performance for the task of Jigsaw Unintended Bias in Toxicity Classi-

fication<sup>1</sup> on Kaggle, which uses the same dataset with SemEval-2021 Task 5, with 93.64% F1-score. We use two models for our QA approach system, and the overview of the system with training and testing phases is described in Figure 4.

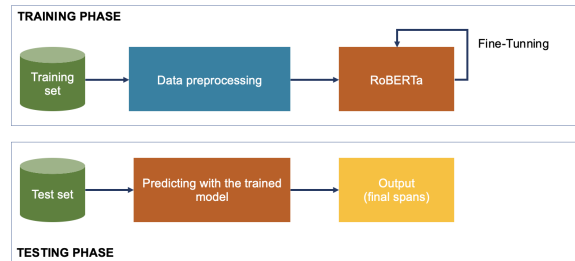


Figure 4: Training and testing phases of toxic spans detection with RoBERTa based system.



Figure 5: Data preprocessing of RoBERTa based system for toxic spans detection.

Firstly, we preprocess the training set with techniques to get the right format for the RoBERTa model, mentioned in Figure 5. The model we used only approve one spans, but several examples have more than one in the training set, and we called it "multi-span". Hence, we split multi-span (\*) into single spans (\*\*) (\*\*\*) as below.

- Plain text:
  - (\*) This bitch is so fucking idiot.
- After splitting:
  - (\*\*) This **bitch** is so.
  - (\*\*\*) This is so **fucking idiot**.

After splitting texts, we tokenize the dataset with a subword model as Byte-Pair Encoding (BPE) (Sennrich et al., 2016). Then, we feed the data into a pre-trained RoBERTa model and fine-tune it with suitable parameters. We analyze the length of the texts in the dataset and set max\_length=512 and epochs=5 for the model. After searching for extensive hyper-parameters, we set the learning\_rate and drop\_out equal to 3e-5 and 0.1, respectively. We also train the model with 5-fold cross-validation. After the training phase, the trained RoBERTa model is used for predicting new toxic spans.

<sup>1</sup><https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification>

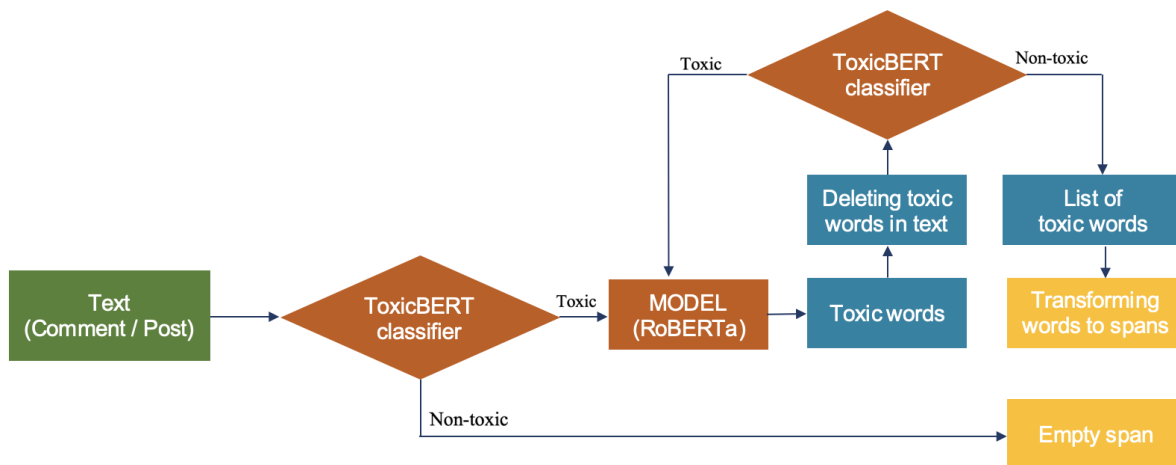


Figure 6: Predicting toxic spans with trained model by QA approach.

In the testing phase, besides using RoBERTa, we use another transfer learning model is ToxicBERT (Hanu and Unitary team, 2020) for identifying toxic comments. With ToxicBERT, we classify the input text into toxic or non-toxic labels before predicting spans. If the result is non-toxic, we stop the prediction, and the result is an empty spans. If it is toxic, we feed the text into the RoBERTa model to predict toxic words. After having the spans, to ensure that the text still has toxic words, we remove the predicted toxic word(s) from the processing text and then recheck its toxicity by ToxicBERT and re-predict its remaining toxic words (if any).

Because final results are words, we transform them into spans for the requirement of this task.

## 4.2 NER Approach Based on spaCy’s Library

In this approach, we tag all the characters spans with text as TOXIC to train the model, and we predict all TOXIC tags in the text set of texts.

For solving this, we choose version 2.2.5 of spaCy’s NER Model (Honnibal and Montani, 2017) because of its exceptionally efficient statistical system in both speed and accuracy for this named-entity recognition. Apart from default entities such as location, person, organization, and so on, spaCy also enables training the model with new entities by updating it with newer examples.

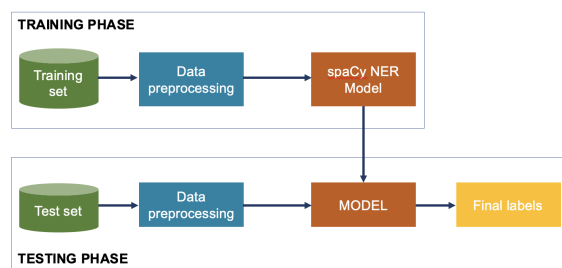


Figure 7: Training and testing phases of toxic spans detection with spaCy based system.

The above Figure 7 shows the process of our spaCy based system. Both training and test sets have to be tokenized before feeding them into the spaCy NER model or being predicted by the TOXIC entities. For more details, in the training phase, the input data have to be in the right format for the spaCy NER model as in the following Figure 8.

spans	text
[7,8,9,10,11,12]	Pretty <b>damned</b> eloquent ... :
[0,1,2,3,4,5,6,7,8,9,15,16,19,20,21,22,23,24,98,99,100,101,102,103,104,105,106]	<b>Nauseating</b> and <b>disgusting</b> . Thank... demonstrate their <b>stupidity</b> .
[]	Not if they shoot you first...

↓  
SpaCy's Format

```
[(Pretty damned eloquent . . .), {'entities': [(7, 12, 'TOXIC')]}],
('Nauseating and disgusting. Thank goodness the... stupidity', {'entities': [(0, 9, 'TOXIC'), (15, 24, 'TOXIC'), (98, 106, 'TOXIC')]}), ('Not if they shoot you first...', {'entities': []})]
```

Figure 8: Process of re-formatting data for spaCy based system.

SpaCy has not published the architecture of their models yet, but they do have a brief explanation about how their models work, especially the NER model, through a four-step formula: embed, encode, attend, and predict.

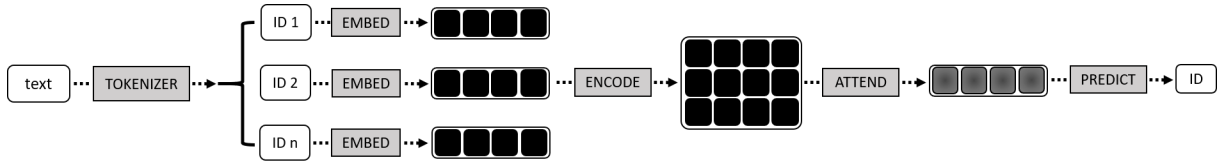


Figure 9: Diagram of how spaCy’s models work.

As in Figure 9, spaCy’s model is fed with unique numerical values (ID) which address a token of a corpus or a class of the NLP task (named entity class). In the first embed stage, word similarities are revealed by extracting hash, which is collected by extracting word features as the lower case, the prefix, the suffix, and the shape. The encode stage is fed with a sequence of word vectors from the previous stage to calculate a representation which is named sentence matrix. In the sentence matrix, the meaning of each token in the context of neighboring tokens is represented in each row, and this is done by using a bidirectional RNN (Schuster and Paliwal, 1997). The output matrix from the second stage is injected into the Attention Layer of the CNN after summarized by a query of vectors. Finally, to predict the toxic class, a softmax function is utilized. After the model is trained, the CNN model is now used for the NER task to extract the toxic class.

The given toxic spans dataset is fed into spaCy’s library for training with a suitable format. During the contest, my team was using spaCy’s library for a small model for English (en\_core\_web\_sm) at version 2.2.5, and we tried different parameters to get the optimal result. When training, the dataset is shuffled and passed through spaCy’s training algorithm in batches with an increment of batch sizes from 4.0 to 32.0 and step of 1.001. Moreover, the drop rate is consistently at 0.5, and most of the experiments loop 45 times.

## 5 Experiments

After building two such systems, we start to experiment on the test set, and the following subsections discuss our results.

### 5.1 Evaluation Metrics

Before going through experimental results, we first discuss the evaluation metrics used in this SemEval-2021 Task 5.

In this task, all of the responding systems from participants are evaluated by **F1** score (Da San Martino et al., 2019). Assuming the system  $S_i$  returns

$C_{S_i}^t$ , which is a toxic character offsets of the post. Let  $G^t$  be the character offsets of the ground truth annotation of  $\mathbf{t}$ . In the following formulas, the **F1** score of system  $S_i$  is computed regarding ground truth  $\mathbf{G}$  of post  $\mathbf{t}$  ( $|\cdot|$  indicates set cardinality).

$$F_1^t(S_i, G) = \frac{2 \cdot P^t(S_i, G) \cdot R^t(S_i, G)}{P^t(S_i, G) + R^t(S_i, G)} \quad (2)$$

$$P_1^t(S_i, G) = \frac{|C_{S_i}^t \cap C_G^t|}{|C_{S_i}^t|} \quad (3)$$

$$R_1^t(S_i, G) = \frac{|C_{S_i}^t \cap C_G^t|}{|C_G^t|} \quad (4)$$

If  $S_G^t$  is empty for posts  $\mathbf{t}$ , we set  $F_1^t(S_i, G) = 1$  and if  $S_{A_i}^t$  is empty,  $F_1^t(S_i, G) = 0$ . Finally, we calculate average of  $F_1^t(S_i, G)$  of all over the posts  $\mathbf{t}$  if test set to get a single **F1** score of the system  $S_i$ .

### 5.2 Experimental Results

The results of our systems compared with other teams’ are shown in Table 2.

Rank	Team name	F1-score
1	HITSZ-HLT	70.83
2	S-NLP	70.77
33	lz1904	67.00
34	UIT-E10dot3	spaCy 66.99
		RoBERTa 52.12
		17.00±1.

Table 2: The results of our systems compared with other teams by F1-score (%).

During the SemEval-2021 Task 5, with the spaCy base system, we achieved rank 34 out of 91 teams, and in the table above, we have shown our result with the spaCy based system and the RoBERTa based system in comparison with rank 1, 2, 33 and random baseline of this task. The F1-score of our best system is 66.99%, 3.84% lower than the first rank team, and 49.09% higher than the baseline model.

### 5.3 Result Analyses

After analyzing our most effective system based on spaCy’s library, we spot crucial errors in predicting and datasets by comparing predicted spans to gold spans. Several records in the given data are standing alone without the context that leads to confusing or multi-meaning. Moreover, comments are using slang(s) or idiom(s), causing null output for our system. We also realize a lack of consistency or highlighting non-toxic spans when annotating data about the datasets. Likewise, several words in the text have spelling mistakes that intentionally also impair our system performance. Evidence for those errors are in Table 3, Appendix.

## 6 Conclusion and Future Work

In this paper, we introduced two proposed systems for toxic spans detection based on named entity and question-answering approaches. We obtained the highest results with the SpaCy’s library based system with the F1-score of **66.99%** and ranked 34 out of 91 teams in SemEval-2021 Task 5.

In future, we plan to improve our systems by implementing various SOTA models for toxic spans detection. With the built systems, we can create friendly online conversations and make social media forums safer for users.

### Acknowledgement

We would like to express our gratitude for the helpful reviews from reviewers.

### References

Valerio Basile, Cristina Bosco, Elisabetta Fersini, Nozza Debora, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, Manuela Sanguinetti, et al. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*

*Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).

Laura Hanu and Unitary team. 2020. Detoxify. Github. <https://github.com/unitaryai/detoxify>.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50.

João Augusto Leite, Diego Silva, Kalina Bontcheva, and Carolina Scarton. 2020. [Toxic language detection in social media for Brazilian Portuguese: New dataset and multilingual analysis](#). In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 914–924, Suzhou, China. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shruthi Mohan, Apala Guha, Michael Harris, Fred Popowich, Ashley Schuster, and Chris Priebe. 2017. The impact of toxic language on the health of reddit communities. In *Canadian Conference on Artificial Intelligence*, pages 51–56. Springer.

Jihyung Moon, Won Ik Cho, and Junbum Lee. 2020. [BEEP! Korean corpus of online news comments for toxic speech detection](#). In *Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media*, pages 25–31, Online. Association for Computational Linguistics.

Luan Thanh Nguyen, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. 2021. [Constructive and toxic speech detection for open-domain social media comments in vietnamese](#).

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.



- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural machine translation of rare words with subword units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Sergey Smetanin. 2020. [Toxic comments detection in russian](#). In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference “Dialogue 2020”*.
- Lynne Tirrell. 2018. Toxic speech: Inoculations and antidotes. *The Southern Journal of Philosophy*, 56:116–144.
- Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffensEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [Semeval-2020 task 12: Multilingual offensive language identification in social media \(offenseval 2020\)](#).

No.	Predicted by spaCy base system	Ground Truth
<i>Requiring context (♠)</i>		
1	No, my poor benighted correspondent, your reductionism reveals an archaic mindset. Intersex, questioning, nongendered, and a vast variety of others have rights too!	No, my poor <b>benighted</b> correspondent, your reductionism reveals an archaic mindset. Intersex, questioning, nongendered, and a vast variety of others have rights too!
<i>Slang(s) / Idiom(s)</i>		
2	Lord have Mercy on us, Trump is running amok.	Lord have Mercy on us, Trump is running <b>amok</b> .
3	No, dum dum-you dont get it. The business does not get to decide who he serves and who he does not. If he makes cakes for one, he must make them for ALL	No, <b>dum dum</b> -you dont get it. The business does not get to decide who he serves and who he does not. If he makes cakes for one, he must make them for ALL
4	Ugh, the worst, burn them at the stake.	Ugh, the worst, <b>burn them at the stake</b> .
<i>Lacking of consistency or highlighting non-toxic spans when annotating (♣)</i>		
5	There is even a website where sore loser Democrats can pretend that Hillary won. La-la-land indeed! #liberaltears #salt #schadenfreude	There is even a website where sore <b>loser</b> Democrats can pretend that Hillary won. La-la-land indeed! #liberaltears #salt #schadenfreude
6	Hey Repubes, I bet you think that "wall" is going to be built on Mexicos dollar too huh? What a bunch of gullible <b>losers</b> !!!!	Hey Repubes, I bet you think that "wall" is going to be built on Mexicos dollar too huh? What a bunch of gullible losers!!!!
7	<Trump was a draft dodger during Vietnam.>The president is a coward and a hypocrite who not only dodged the draft and avoided any and every opportunity to serve... No one has more contempt for the flag than does Donald J. Trump. No one.	<Trump was a draft dodger during <b>Vietnam</b> .>The president is a coward and a hypocrite who not only dodged the draft and avoided any and every opportunity to serve... No one has more contempt for the flag than does Donald J. Trump. No one.
<i>Spelling mistake</i>		
8	FUKC YOU @QuinHillyer YOU ARE THE ONE NEED TO GO, YOU ANTI AMERICA FAKE MOTHER FUKCER	<b>FUKC YOU</b> @QuinHillyer YOU ARE THE ONE NEED TO GO, YOU ANTI AMERICA <b>FAKE MOTHER FUKCER</b>
9	Hey Repubes, I bet you think that "wall" is going to be built on Mexicos dollar too huh? What a bunch of gullible <b>losers</b> !!!!	Hey Repubes, I bet you think that "wall" is going to be built on Mexicos dollar too huh? What a bunch of gullible losers!!!!

Table 3: Examples for result analyses.

(♠) The table shows that with this example, our system predicts that there are no toxic spans; meanwhile, in ground truth, the word **benighted** is highlighted to be toxic. We assume that because of lacking the context of the text and the word **benighted** also has multi-meaning.

(♣) The table shows that in example No. 5, the word **loser** is annotated to be toxic when in example No. 6, also having the plural form of word **loser** but not to be highlighted. Meanwhile, in example No. 7, the spans **Vietnam.>** is highlighted even if it does not have toxicity.

# SkoltechNLP at SemEval-2021 Task 5: Leveraging Sentence-level Pre-training for Toxic Span Detection

David Dale<sup>‡</sup>, Igor Markov<sup>‡,†</sup>, Varvara Logacheva<sup>‡</sup>, Olga Kozlova<sup>†</sup>,  
Nikita Semenov<sup>†</sup>, and Alexander Panchenko<sup>‡</sup>

<sup>‡</sup>Skolkovo Institute of Science and Technology, Moscow, Russia

<sup>†</sup>Mobile TeleSystems (MTS), Moscow, Russia

{d.dale,igor.markov,v.logacheva,a.panchenko}@skoltech.ru  
{oskozlo9,nikita.semenov}@mts.ru

## Abstract

This work describes the participation of the Skoltech NLP group team (Sk) in the Toxic Spans Detection task at SemEval-2021. The goal of the task is to identify the most toxic fragments of a given sentence, which is a binary sequence tagging problem. We show that fine-tuning a RoBERTa model for this problem is a strong baseline. This baseline can be further improved by pre-training the RoBERTa model on a large dataset labeled for toxicity at the sentence level. While our solution scored among the top 20% participating models, it is only 2 points below the best result. This suggests the viability of our approach.

## 1 Introduction

Toxicity and offensive content is a major concern for many platforms on the Internet. Therefore, the task of toxicity detection has attracted much attention in the NLP community (Wulczyn et al., 2017; Hosseini et al., 2017; Dixon et al., 2018). Until recently, the majority of research on toxicity focused on classifying entire user messages as toxic or safe. However, the surge of work on text detoxification, i.e., editing of text to keep its content and remove toxicity (Nogueira dos Santos et al., 2018; Tran et al., 2020), suggests that localizing toxicity within a sentence is also useful. If we know which words of a sentence are toxic, it is easier to “fix” this sentence by removing or replacing them with non-toxic synonyms. Mathew et al. (2020) make human labelers annotate the spans as rationales for classifying a comment as hateful, offensive, or normal. They show that using such spans when training a toxicity classifier improves its accuracy and explainability and reduces unintended bias towards toxicity targets.

This year the SemEval hosts the first competition on toxic spans detection, namely, SemEval-2021

Task 5<sup>1</sup> (Pavlopoulos et al., 2021). It provides training, development, and test data for English. As far as we know, it is the first attempt to explicitly formulate toxicity detection as sequence labeling instead of classification of sentences.

Multiple NLP tasks recently benefited from transfer learning — transfer of probability distributions learned on some task to another model solving a different task. The most common example of transfer learning is the use of embeddings and language models pre-trained on unlabeled data (e.g. ELMo (Peters et al., 2018), BERT (Devlin et al., 2019) and its variations, T5 (Raffel et al., 2020), etc.) on other tasks (e.g. He et al. (2020); Wang et al. (2020) *inter alia* use pre-trained BERT models to perform tasks from the GLUE benchmark (Wang et al., 2018)).

Word-level toxicity classification can be formulated as a sequence labeling task, which also actively uses the pre-trained models mentioned above. BERT comprises the versatile information on words and their context, which allows to successfully use it for sequence labeling tasks of different levels: part-of-speech tagging and syntactic parsing (Koto et al., 2020), named entity recognition (Hakala and Pyysalo, 2019), semantic role labeling (He et al., 2019), detection of Machine Translation errors (Moura et al., 2020).

This diversity of applications suggests that word-level toxicity detection can also benefit from pre-trained models. Besides that, toxicity itself has been successfully tackled with BERT-based models. Research on sentence-level toxicity extensively used BERT and other pre-trained models. Both language-specific and multilingual BERT models were used to fine-tune toxicity classifiers (Leite et al., 2020; Ozler et al., 2020). This shows that BERT has information on toxicity.

<sup>1</sup><https://competitions.codalab.org/competitions/25623>

Thus, we follow this line of work. Namely, we fine-tune a RoBERTa model (Liu et al., 2019) to perform a sequence labeling task. Besides that, we train a model for sentence classification on the Jigsaw dataset of toxic comments and use the information from this model to detect toxicity at the subsentential level. This helps us overcome the insufficient data size.

This is in line with previous work, which has shown that sentence-level labels can be used in combination with token labels (Rei and Søgaard, 2019) or completely substitute them (Rei and Søgaard, 2018; Schmaltz, 2019).

In our experiments, we test the hypothesis that the sentence-level toxicity labeling can be used for a sequence labeler that recognizes toxic spans in text. We suggest three ways of incorporating this data: as a corpus for pre-training, pseudo-labeling, and for joint training of sentence-level and token-level toxicity detection models. Our experiments show that the latter method yields the best result. Moreover, we show that using sentence-level labels can dramatically improve toxic span prediction when the dataset with token-level labels is small.

The contributions of this work are the following:

- We successfully use the dataset labeled for toxicity at the sentence level for token-level toxicity labeling,
- We propose a model for joint sentence- and token-level toxicity detection,
- We analyze the performance of our models, showing their limitations and reveal the ambiguities in the data.

## 2 The task

The training data of the task comprises 7,940 English comments with character-level annotations of toxic spans. The labeling was performed manually by crowd workers.

The spans labeled as toxic often contain rude words: “*Because he’s a moron and a bigot. It’s not any more complicated than that.*” (toxic spans are underlined). Other toxic spans consist of words that become toxic in context: “*Section 160 should also be amended to include sexual acts with animals not involving penetration”.* Borders of some toxic spans fall in the middle of a word; we treat such cases as markup errors.

As a development set, we use the trial dataset of 690 texts provided by the task organizers. We

evaluate our final models on the hidden test set of the task consisting of 2,000 texts.

## 3 Pre-training for toxic span detection

Here we give the motivation behind our models and describe their architecture and training setup.

### 3.1 Motivation

Our intuition is that the toxicity is often lexically-based, i.e., there are certain words that are considered offensive and make the whole sentence toxic. In this case, we expect that as we add extra data to our toxic span dataset, after some point, the vocabulary of toxic words in it will saturate and stop increasing. However, Figure 1 shows that the size of the toxic vocabulary linearly depends on the dataset size, which suggests that its size is insufficient for the task. In this case, the model will often need to label unseen words. To mitigate the lack of data, we leverage the additional dataset with toxicity information, namely, the Jigsaw toxic comments dataset<sup>2</sup> which features 140,000 user utterances labeled as toxic or safe.

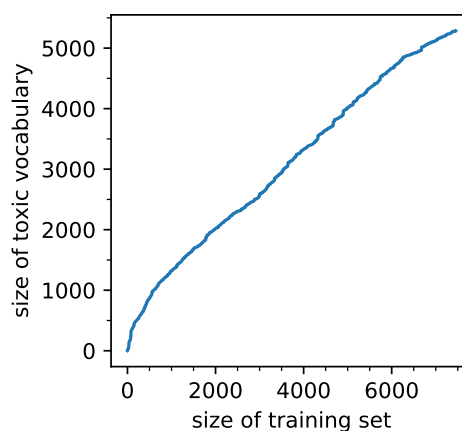


Figure 1: Size of the toxic vocabulary as a function of the corpus size.

### 3.2 Transfer learning for spans detection

Our base model is RoBERTa. We fine-tune roberta-base model on the toxic spans training set for sequence labeling task (further denoted as **RoBERTa tagger**). This model already gives promising results. We further improve it by providing it with the additional training signal from the

<sup>2</sup><https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data>

Jigsaw toxic comments dataset. We propose three ways of incorporating this data.

The first way is **pseudo-labeling** (Lee et al., 2013). We apply the **RoBERTa tagger** to predict the toxic spans in the Jigsaw dataset. We use these predictions to further train the model.

Another option is to use the Jigsaw data to fine-tune RoBERTa with it. We suggest two scenarios. The first is to fine-tune the model on the Jigsaw dataset for the sentence classification task, and then on the toxic spans dataset — this model is referred to as **RoBERTa classifier + tagger**. In this case, the model has different output layers for the two tasks, and other layers are shared.

Finally, we propose a novel architecture for joint token- and sentence-level classification, where the score  $\mathbf{y}$  for a sentence  $\mathbf{x} = \{x_1, x_2, \dots, x_n\}$  is computed as the average of word-level scores:

$$\hat{\mathbf{y}} = \sigma \left( \alpha + \beta \frac{1}{n} \sum_{i=1}^n \hat{y}_i^\gamma \right)$$

where  $\alpha$ ,  $\beta$  and  $\gamma$  are trainable parameters, and  $\sigma$  is the logistic function. This model does not need to be trained on the data with token-level labeling but can get token-level toxicity information from sentence labels. We fine-tune this model both on Jigsaw and toxic spans datasets. The model is referred to as **tagging classifier**.

### 3.3 Working with spans

To reformulate toxic spans detection as a token classification problem, we label a token as toxic if at least one of its characters is toxic. When projecting the predicted token-level labels back to the character level, we try two strategies:

1. Consider a token to be toxic if its toxicity score is higher than the threshold, do not force the labels of tokens within a word to agree with each other.
2. Consider a word to be toxic if the aggregated toxicity score of all its tokens is higher than the threshold. We try four different aggregation functions: min, max, mean, and the simplified naive Bayes formula:

$$\tilde{x} = \frac{\prod_i x_i}{\prod_i x_i + \prod_i (1 - x_i)}.$$

In both methods, we label a space character as toxic only if the characters both to the right and to the left of it are toxic.

## 4 Baselines

In this section, we present a set of common baseline approaches used for sequence tagging, such as CRF and LSTM with pretrained word embeddings. We implement them in order to analyze the performance of our methods in the context of other techniques.

**Word-based LogReg** This is a vocabulary-based method: we label words as toxic if they appear in our toxic vocabulary. The vocabulary is created as follows. We create a set of toxic and safe phrases, where toxic phrases are toxic spans from our data and safe phrases are sentences from our data with removed toxic spans. We then train a binary logistic regression classifier of toxic and safe phrases using words as features. The by-product of this classifier is the list of weights for all words from the data. We consider words with weights greater than a threshold as toxic.

**Attention-based LogReg** Another approach to represent words is to take their attention weights from a RoBERTa-based sentence-level toxicity classifier (we train it on the Jigsaw dataset). We assemble attention weights from all RoBERTa heads and layers in a single vector of dimension 144. These vectors are used as features in a logistic regression classifier. This approach is motivated by the fact that a RoBERTa model trained to recognize toxicity puts more emphasis on certain words associated with sentence-level toxicity. Surprisingly, this model underperforms the logistic regression classifier, which uses words as features.

**Conditional Random Fields** We suggest that the toxicity level of a word can be context-dependent, so we also experiment with sequence labeling models. We try Conditional Random Fields (CRF) (Lafferty et al., 2001) model. It uses the following features: the word itself, the word’s part of speech, whether the word is a digit and consists of uppercase letters. Each word is represented with these features of the current, previous, and next words. The model performs closely to the attention-based classifier.

**Sequence labeling with LSTM** Finally, we experiment with the LSTM architecture (Hochreiter and Schmidhuber, 1997). We implement a Bi-LSTM network and also train an LSTM tagger from the AllenNLP library.<sup>3</sup> We do not use any pre-

<sup>3</sup><https://github.com/allenai/allennlp>

trained embeddings in the Bi-LSTM model and use two versions of the AllenNLP LSTM: without pre-trained embeddings and with GloVe embeddings (Pennington et al., 2014).

## 5 Evaluation

### 5.1 Experimental Setting

For each transfer learning model, we use two-stage fine-tuning. We first train only the output layers of the models with the learning rate of  $10^{-3}$ , and then the whole models with the learning rate of  $10^{-5}$ . In both cases, we use linear learning rate warm-up for 3000 steps. We use the AdamW optimizer (Loshchilov and Hutter, 2019) and the batch size of 8, and early stopping to determine the number of training steps. We use the `transformers`<sup>4</sup> library for training.

For all the models which use the additional data from Jigsaw, we apply this scheme twice. The **pseudo-label** model is first fine-tuned on the original toxic spans dataset, and then on the self-labeled Jigsaw dataset, whereas the **RoBERTa classifier + tagger** and **tagging classifier** are first fine-tuned on the Jigsaw dataset (as classifiers), and then on the toxic spans dataset (as taggers).

### 5.2 Results

The scores of our models and the competing systems are shown in Table 1. Our best submitted system (**tagging classifier**) had the  $F_1$ -score of 0.681 on the test set, while the best team over the whole task got 0.708. This brings our team to the top 20% of the leaderboard. The pseudo-labeling approach was only marginally worse, scoring 0.674. Simply fine-tuning RoBERTa only on the tagging problem scored 0.668. On the other hand, none of our baselines could approach this result. Our best baseline is the word-based LogReg classifier. Apparently, other models fail to learn even the toxic vocabulary because their word representations are not informative enough.

While our best-performing model is only 18th best out of 92 participating systems, the results of the top systems are fairly close to ours (the difference is less than 2.5%). The variation of deep learning models often falls in this margin (Reimers and Gurevych, 2017). For our models, the sample standard deviation of the  $F_1$ -score is about 0.9%, so the difference between their performance is likely to be statistically insignificant.

<sup>4</sup><https://huggingface.co/transformers>

Model	$F_1$ score
<b>Top-5 participants</b>	
HITSZ-HLT	0.708
S-NLP	0.707
hitmi&t	0.698
L	0.698
YNU-HPCC	0.696
<b>Our models</b>	
tagging classifier	0.683
pseudo-labeling	0.682
RoBERTa tagger	0.678
RoBERTa classifier + tagger	0.670
<b>Our baselines</b>	
Word-based LogReg	0.556
LSTM basic embeddings	0.538
Bi-LSTM basic embeddings	0.530
Attention-based Logreg	0.524
CRF	0.523
LSTM Glove embeddings	0.497

Table 1: Performance of our models (baselines and RoBERTa-based models) and their comparison with the 5 best-performing participants. Models within each section are sorted from best to worst.

An important hyper-parameter of the models is the probability threshold. It is usually fine-tuned on the development set. However, the development set provided for the task is too small. The threshold fine-tuned on it performs even worse than the standard threshold of 0.5. Thus, during the evaluation period of the competition, we tried submitting models with different threshold values. While this is not a completely fair practice because we indirectly used the test for tuning a model parameter, we suspect that many teams were overfitting to the test set in a similar way. We suggest that in order to make the evaluation fair, the results of the models on the final test set should not be available before the end of the competition, even in the indirect way (i.e., in the form of teams ranking without scores as it was done in this competition).

The best results which we report here were achieved with the threshold of 0.6 (see Table 1). We compare these results with those of the same models with the default threshold of 0.5 in Table 3. It shows that these scores are lower by up to 1%.

Another hyper-parameter of our models is the method of converting token-level labels to

Aggregation Type	F <sub>1</sub> score
No aggregation	0.685
<b>Aggregation of word-level scores</b>	
max token score	0.673
min token score	0.641
average token scores	0.670
naive Bayes	0.653

Table 2: Scores of the **tagging classifier** model with different token aggregation methods (computed on the development set).

Model	threshold	
	0.5	0.6
tagging classifier	0.681	<b>0.683</b>
pseudo-labeling	0.674	0.682
RoBERTa tagger	0.668	0.678
RoBERTa classifier + tagger	0.664	0.670

Table 3: F<sub>1</sub>-scores of models with different probability thresholds.

character-level labels. We compare different methods on the development set (see Table 2). Surprisingly, the prediction of labels for each token individually with no aggregation works better than assigning labels to the whole words. This might happen because the attempts to decode words consistently lead to the propagation of wrongly predicted labels. Following this observation, we use the no-aggregation strategy for all models.

### 5.3 Efficiency of pre-training

To understand the effect of the use of additional sentence-labeled data, we compare the performance of **RoBERTa tagger** (a model which uses only the toxic span dataset) and **tagging classifier** (a model which uses sentence-labeled Jigsaw data in addition to the toxic span dataset) models trained on subsets of the data of different sizes. We would like to see if the usefulness of additional sentence-labeled data reduces as we get more data with token-level labeling.

Figure 2 plots the F<sub>1</sub>-scores of the two models trained on datasets of sizes between 10 and 7,940 sentences. It shows that when the training set size is between 10 and 1,000, pre-training with sentence-level annotations gives a considerable boost in performance. However, the effect of this pre-training becomes insignificant after the size of the data with

word-level labeling reaches around 3,000. Thus, this pre-training strategy is efficient only in cases when the size of the data with word-level labeling is very small.

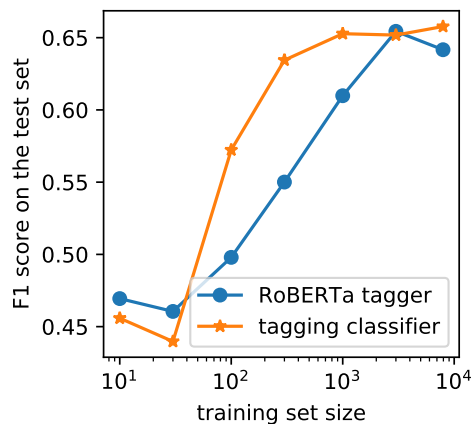


Figure 2: Learning curves for two transfer learning models, with (**tagging classifier**) and without (**RoBERTa tagger**) additional sentence-level data.

### 5.4 Error analysis

We analyze the errors of our best submitted system (**tagging classifier**) by comparing its predictions with the ground truth labels released after the end of the competition.

The vocabulary of false negative spans (527 unique tokens) is more diverse than that of false positives (275 unique tokens), while the number of false positives and false negatives in the test set is comparable (860 vs 813 tokens). It may indicate that the model is cautious and prefers to highlight only the hypotheses which have high confidence, while human annotators are more creative in their analysis. We give some examples of correct and incorrect labelings by our model in Table 4.

The most frequent false positive words characterize incompetence or lack of mental capacities: *stupid, idiot, ignorant, moron, dumb*, etc. Other frequent false positives are derogatory (*pathetic, ridiculous, ass, garbage, loser*, etc.), denounce particular misdeeds (*liar, troll, racist, hypocrite*, etc.), or express general negativity (*damn, fuck*, etc.). It is not obvious why human annotators label them as toxic in some cases, and as non-toxic in other cases. We suspect that inter-annotator agreement on such words is not very high.

The most frequent false negative words are function words: *and, the, are, a, you* etc. It happens because annotators sometimes label the whole text

Correct labeling
See a shrink you <u>pathetic troll</u> . They're not patriots. They're <u>vandals</u> , <u>thieves</u> , and <u>bullies</u> . Trudeau and Morneau are fiscally and economically <u>inept</u> and <u>incompetent</u> .
Incorrect labeling
That's right. They are not normal. And I am starting from the premise that they are <u>ABNORMAL</u> . Proceed with the typical racist, <u>bigot</u> , <u>sexist rubbish</u> . Thanks! ADN is endorsing, without officially endorsing. <u>Bunch of cowards!!!</u> Rabidly <u>anti-Canadian troll</u> .

Table 4: Examples of ground truth (underlined) and predicted (**in bold**) toxic spans

Top 20 false positive words		Top 20 false negative words	
stupid	ass	and	of
idiot	liar	the	have
ignorant	garbage	are	loser
moron	loser	a	crap
dumb	fools	ignorant	all
idiots	troll	racist	chemical
fool	crap	you	that
pathetic	damn	in	not
stupidity	fuck	is	bunch
ridiculous	clown	to	dumb

Table 5: The most common false positive and false negative words

or a large chunk of it as toxic. The more meaningful false negatives belong to the same classes as the false positives (*ignorant*, *racist*, *loser*, etc.). The most common false positive and false negative words are listed in Table 5.

In general, the performance on this task might be limited to 0.7  $F_1$ -score (the quality of the best-performing model) by the ambiguity of the annotations. In future work, it

## 6 Conclusions

We present a number of models for the detection of toxic spans within toxic sentences. All models are RoBERTa language models fine-tuned on the data with character-level labeling of toxic spans. In addition to that, we perform fine-tuning on an additional dataset with sentence-level toxicity labeling. This yields an improvement. However, our analysis shows that the effect of such pre-training is marginal when the main dataset size exceeds 1,000 samples. Therefore, substantial improvement is observed for small dataset sizes. Nevertheless, the models we propose can be useful in extremely

low-resource scenarios.

Our model performs closely to the winning systems. We suggest that the differences between the 20 top models might be attributed to the variation of deep learning models and overfitting the test set. In addition to that, the error analysis shows that some errors in our model might be due to inconsistencies in the test data.

We release the code required to reproduce our experiments online.<sup>5</sup>

## Acknowledgements

This work was conducted in the framework of the joint Skoltech-MTS laboratory.

## References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

<sup>5</sup><https://github.com/skoltech-nlp/toxic-span-detection>



- for *Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73.
- Kai Hakala and Sampo Pyysalo. 2019. **Biomedical named entity recognition with multilingual BERT**. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61, Hong Kong, China. Association for Computational Linguistics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. **DeBERTa: Decoding-enhanced BERT with disentangled attention**. *CoRR*, abs/2006.03654.
- Shexia He, Zuchao Li, and Hai Zhao. 2019. **Syntax-aware multilingual semantic role labeling**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5350–5359, Hong Kong, China. Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. **Deceiving google’s perspective API built for detecting toxic comments**. *CoRR*, abs/1702.08138.
- Fajri Koto, Afshin Rahimi, Jey Han Lau, and Timothy Baldwin. 2020. **IndoLEM and IndoBERT: A benchmark dataset and pre-trained language model for Indonesian NLP**. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 757–770, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of the 18th International Conference on Machine Learning 2001*.
- Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3.
- João Augusto Leite, Diego Silva, Kalina Bontcheva, and Carolina Scarton. 2020. **Toxic language detection in social media for Brazilian Portuguese: New dataset and multilingual analysis**. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 914–924, Suzhou, China. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **Roberta: A robustly optimized BERT pretraining approach**. *CoRR*, abs/1907.11692.
- Ilya Loshchilov and Frank Hutter. 2019. **Decoupled weight decay regularization**. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. **Hatexplain: A benchmark dataset for explainable hate speech detection**. *CoRR*, abs/2012.10289.
- João Moura, miguel vera, Daan van Stigt, Fabio Kepler, and André F. T. Martins. 2020. **Ist-unnabel participation in the wmt20 quality estimation shared task**. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1029–1036, Online. Association for Computational Linguistics.
- Kadir Bulut Ozler, Kate Kenski, Steve Rains, Yotam Shmargad, Kevin Coe, and Steven Bethard. 2020. **Fine-tuning for multi-domain and multi-label uncivil language detection**. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 28–33, Online. Association for Computational Linguistics.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. **Deep contextualized word representations**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. **Exploring the limits of transfer learning with a unified text-to-text transformer**. *Journal of Machine Learning Research*, 21(140):1–67.

- Marek Rei and Anders Søgaard. 2018. [Zero-shot sequence labeling: Transferring knowledge from sentences to tokens](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302, New Orleans, Louisiana. Association for Computational Linguistics.
- Marek Rei and Anders Søgaard. 2019. Jointly learning to label sentences and tokens. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6916–6923.
- Nils Reimers and Iryna Gurevych. 2017. [Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.
- Cicero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. [Fighting offensive language on social media with unsupervised text style transfer](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 189–194, Melbourne, Australia. Association for Computational Linguistics.
- Allen Schmaltz. 2019. Detecting local insights from global labels: Supervised & zero-shot sequence labeling via a convolutional decomposition. *arXiv preprint arXiv:1906.01154*.
- Minh Tran, Yipeng Zhang, and Mohammad Soleymani. 2020. [Towards a friendly online community: An unsupervised style transfer framework for profanity redaction](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2107–2114, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.
- Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. [Structbert: Incorporating language structures into pre-training for deep language understanding](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.

# Entity at SemEval-2021 Task 5: Weakly Supervised Token Labelling for Toxic Spans Detection

**Vaibhav Jain**

New Delhi, India

vaibhav29498@gmail.com

**Mina Naghshnejad**

Oakland, CA, USA

mina.naghshnejad@gmail.com

## Abstract

Detection of toxic spans - detecting toxicity of contents in the granularity of tokens - is crucial for effective moderation of online discussions. The baseline approach for this problem using the transformer model is to add a token classification head to the language model and fine-tune the layers with the token labeled dataset. One of the limitations of such a baseline approach is the scarcity of labeled data. To improve the results, We studied leveraging existing public datasets for a related but different task of entire comment/sentence classification. We propose two approaches: the first approach fine-tunes transformer models that are pre-trained on sentence classification samples. In the second approach, we perform weak supervision with soft attention to learn token level labels from sentence labels. Our experiments show improvements in the F1 score over the baseline approach. The implementation has been released publicly.<sup>1</sup>

## 1 Introduction

The growth of social media platforms has led to an increase in hate speech and abusive language in online communities, primarily due to the anonymity provided on such platforms (Mollas et al., 2020). Since manual moderation is not feasible for the gigantic amount of textual data, automated toxicity detection has received significant attention with numerous datasets being released in recent years (Pavlopoulos et al., 2020). However, most of the existing work on toxicity detection labels the entire comment as toxic or non-toxic and does not provide information about which specific part of the comment is toxic. In practice, human moderators (e.g., news portals moderators) can benefit from information on which character indices of the

part of the comment that is toxic instead of just a system-generated unexplained toxicity score per post. Designing models that can accurately locate toxic spans within a text is thus a crucial step towards successful semi-automated moderation. This is challenging because of the scarcity of datasets that are labeled on a segment or token level.

This paper explains our approaches for the SemEval-2021 Task 5 which requires us to identify the character offsets for the toxic spans within a comment (Pavlopoulos et al., 2021). We explore possible techniques for improving the results of the vanilla transformer model by leveraging several available public datasets.

## 2 Related Work and Background

**Sequential adaptation** Using pre-trained language models has recently proved to be effective for language understanding tasks (Phang et al., 2018). The supplementary training is particularly beneficial when labeled data is scarce (Phang et al., 2018). A popular transfer learning technique in Natural Language Processing (NLP) has been to pre-train sentence encoder neural networks, such as BERT (Devlin et al., 2019), on unsupervised tasks and then fine-tune the encoders for the target supervised learning task. However, this approach can be found inadequate if the input distribution for the target task is considerably different from that of the corpus used for pre-training. Phang et al. (2019) suggested that training on related data-rich supervised tasks as an intermediate step can help in making the final trained model more robust and effective. This approach is called *Supplementary Training on Intermediate Labeled-data Tasks*. This can also help the model in learning the domain knowledge when in-domain data is available.

**Weakly Supervised Learning** Zhou (2018) defined *inexact* supervision as a type of weakly super-

<sup>1</sup><https://github.com/vaibhav29498/Toxic-Spans-Detection>

vised learning in which coarse-grained labels are used to train for a more specific problem. [Rei and Søgaard \(2018\)](#) used this approach for inferring token-level labels by training a sentence classification model. They used a bidirectional LSTM to get a contextual representation  $\tilde{h}_i \in \mathbb{R}^m$  for every token  $w_i$ , and pass it through a fully-connected layer with hyperbolic tangent activation to obtain  $h_i \in \mathbb{R}^n$ :

$$h_i = \tanh(W_h \tilde{h}_i + b_h) \quad (1)$$

where  $W_i \in \mathbb{R}^{n \times m}$  and  $b_i \in \mathbb{R}^n$ .

They then used a two-layer feed forward neural network followed by a soft-attention layer to get a single-valued attention score  $a_i$  for each token:

$$e_i = \tanh(W_c h_i + b_c) \quad (2)$$

$$\tilde{e}_i = W_{\tilde{c}} e_i + b_{\tilde{c}} \quad (3)$$

$$\tilde{a}_i = \sigma(\tilde{e}_i) \quad (4)$$

where  $W_c \in \mathbb{R}^{p \times n}$ ,  $b_c \in \mathbb{R}^p$ ,  $W_{\tilde{c}} \in \mathbb{R}^{1 \times p}$ ,  $b_{\tilde{c}} \in \mathbb{R}$ , and  $\tilde{a}_i$  is the normalized attention score. These scores indicate the importance of the tokens towards predicting the sentence class and can be considered as the token-level predictions. Their normalized forms are used for constructing a weighted average sentence representation  $c$ , which is used to compute the prediction score  $y$  with a value higher than 0.5 indicating a positive class:

$$a_i = \frac{\tilde{a}_i}{\sum_{k=1}^N \tilde{a}_k} \quad (5)$$

$$c = \sum_{i=1}^N a_i h_i \quad (6)$$

$$d = \tanh(W_d c + b_d) \quad (7)$$

$$y = \sigma(W_y d + b_y) \quad (8)$$

where  $W_d \in \mathbb{R}^{q \times n}$ ,  $b_c \in \mathbb{R}^q$ ,  $W_y \in \mathbb{R}^{1 \times q}$ , and  $b_y \in \mathbb{R}$ . The authors used a modified loss function to ensure that the model learns high-quality token labels:

$$L = \sum_j [(y^{(j)} - \tilde{y}^{(j)})^2 + \lambda(\min_i(\tilde{a}_i)^2 + (\max_i(\tilde{a}_i) - \tilde{y}^{(j)})^2)] \quad (9)$$

where  $y^{(j)}$  is the predicted score,  $\tilde{y}^{(j)}$  is the ground-truth,  $\min_i(\tilde{a}_i)$  and  $\max_i(\tilde{a}_i)$  are the lowest and highest attention scores respectively for the  $j^{th}$  sentence.

[Karamanolakis et al. \(2019\)](#) used a modified version of this approach to generate segment labels for text review classification problems. The segment embeddings are generated by feeding the word embeddings into a convolutional neural network. The output of the convolution neural network is passed through a single layer with softmax activation to get the segment-level prediction. Additionally, the word embeddings are passed through a bidirectional GRU network with sigmoid attention to find the attention weights. These weights are used to aggregate the segment-level predictions into a single prediction for the entire review.

### 3 Methodology and Experimental Setup

In this section, we will present our two solutions. Both of our approaches are using BERT pre-trained language model. In the first approach, we first fine-tune the BERT model with additional labeled data explained in Subsection 3.1 and then perform another round of fine-tuning using token classification head with task training data. In the second approach, we apply weak supervision to learn token labels for the additional dataset. We then use the augmented labeled token dataset to fine-tune the token classifier head and use it to predict labels for the task test data set.

#### 3.1 Datasets

As the main data source for training our models, we used SemEval-2021 Task 5 data. Additionally, we used five publicly available English-language datasets for sentence classification to improve our results.

##### 3.1.1 Token-level Labelled Data

We used two token-level labeled datasets, the first being the one provided by the SemEval-2021 Task 5 organizers which are composed of 9,939 English posts along with their toxic spans. The span for a single post is a possible-empty list of character indices that have been marked as toxic by crowd-annotators. The dataset has been divided into training and test sets with 2,000 samples in the latter.

The second dataset used is HateXplain ([Mathew et al., 2020](#)) which is composed of 20,148 tweets and Gab posts, each of which is classified as hateful,

offensive, or normal, by three annotators. If an annotator masks a post as either hateful or offensive, they are also asked to mark the span (*rationale*) which influenced their decision. We considered a token to be toxic if it was included in at least one annotator’s rationale, and excluded all posts with no toxic token, which left us with 11,415 posts.

### 3.1.2 Sentence-level Labelled Data

We used five other datasets which consisted of only sentence-level labels:

- Jigsaw/Conversation AI toxic comment classification challenge dataset<sup>2</sup> - Composed of Wikipedia’s talk page edits, it labels 223,549 posts into zero or more categories of toxicity (toxic, severely toxic, obscene, threat, insult, and identity hate). We bundled each of them into a single category, and 22,468 posts were categorized as toxic.
- Hate speech and offensive language dataset (Davidson et al., 2017) - 24,783 tweets categorized to hate speech, offensive language, or neither. We labeled 20,620 belonging to the former two categories as toxic.
- Online harassment dataset (Golbeck et al., 2017) - Composed of 20,360 tweets out of which 5,285 have been labeled as harassing.
- Impermium dataset for detecting insults in social commentary<sup>3</sup> - 6,594 comments from online forums, out of which 1,742 were identified as insulting to at least one of the participants.
- OffensEval-2020 subtask-A extended test dataset - 5,993 tweets out of which 3,002 have been labeled as offensive.

Merging these five datasets resulted in a collection of 281,279 comments out of which 53,117 were labeled as toxic.

## 3.2 Approach 1: Sequential fine tuning

In approach 1, we first fine-tune the sentence classification model with additional data. The sentence classification model is built upon the base uncased

<sup>2</sup>[www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data](https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data)

<sup>3</sup>[www.kaggle.com/c/detecting-insults-in-social-commentary/data](https://www.kaggle.com/c/detecting-insults-in-social-commentary/data)

BERT model, which maps every sentence to a vector of size 768. A two-layer sentence classifier is added to the BERT base model. The first layer of the head is a linear layer with leaky ReLU activation (negative slope of 0.1) that maps it to a vector of size 64. The second layer of the head is a dense layer that maps 64 nodes to a single-valued prediction score with sigmoid activation. The model is shown in Figure 1.

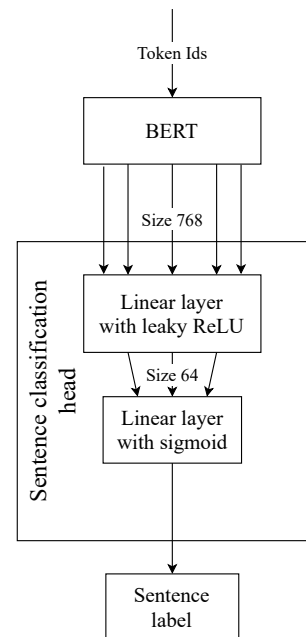


Figure 1: Model for sentence classification

We used the pre-trained BERT language model as the starting point for the BERT model. The layers of sentence classification head were randomly initialized. We applied the WordPiece tokenizer and added the special BERT tokens *[CLS]* and *[SEP]* to each sentence. We removed 5,136 samples as their number of tokens exceeded the base BERT model’s maximum limit (512). The model was trained for a single epoch using the AdamW optimizer (Loshchilov and Hutter, 2019) with a batch size of 16.

The fine-tuned model was used to predict token classes. To predict token classes, a token classification was used. The token classification head architecture is similar to the sentence classification head with two linear nodes. The first layer maps 768-dimensional inputs to 64 nodes and applies the ReLU activation function with the negative slope of 0.1. The second layer maps input to a single token prediction score with a sigmoid activation function.

### 3.3 Approach 2 : Augmenting token labels with weak supervision

In our second approach, we first augment our training set with imperfect labels that we learn by weak supervision. We then use the augmented training set to fine-tune the token classifier model and use it to predict the labels of our test data. The model used for augmenting token labels using sentence-level labeled dataset described in Section 3.1.2, is loosely based upon the solution proposed by Karamanolakis et al. (2019). The model is shown in Figure 2. We first obtain the initial token embeddings from a pre-trained BERT model and input it to the token augmentation network shown in Fig 2. The input is connected on one side to token labels as well as to sentence-level predictions through two BiGRU layers. The GRU layers generate an attention weight for every token. These weights are used for finding a weighted average of the token-level predictions, which is used to calculate the prediction for the entire sentence.

Equation 9 is used as the loss function to train the weights of the model. The sentence labels are calculated from token labels following equations 5 to 8. The loss function optimizes the sentence and token labels. first, it makes sure sentence classifications are closest to the sentence labels. Second, it optimizes token labels by considering the minimum values of token labels in each sentence. It makes sure the minimum label of the tokens in a single sentence is zero to make sure all tokens in a sentence do not have a positive sentiment. It makes sure the most toxic token in the sentence has the same label as the label for the sentence. We trained the model for five epochs using the AdamW optimizer with a batch size of 16.

The token labels generated by the model described above are used as additional training data. The BERT base model is initialized with the publicly available pre-trained version. Only those artificially-generated samples in the sentences with the correctly predicted label (prediction score  $\geq 0.5$ ) are used. We used the prediction scores as the labels for the additional dataset instead of having discrete values of zero and one based on some threshold. We also subtracted the value of 0.1898 from these labels to make their mean equal to that of the labels of the contest dataset.

For generating the character offsets, we considered an entire word as toxic if any single of its subword tokens were identified as toxic.

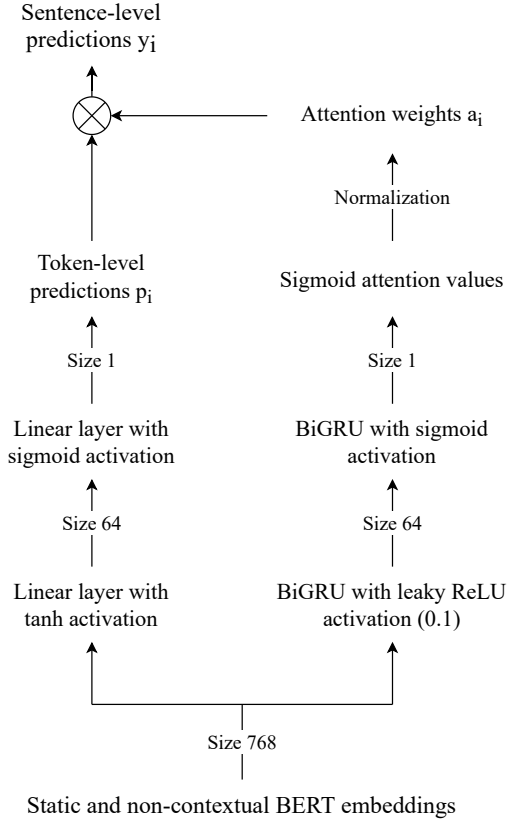


Figure 2: Model for generating token labels

## 4 Results

**Disclaimer:** The section contains offensive, obscene, and hateful content; however this is necessary to showcase the results of this work.

The contest organizers used the average F1 score as the performance metric. For a sample  $S_i$ , if  $\tilde{Y}_i$  and  $Y_i$  are the predicted and actual set of character offsets, then the F1 score  $F_1^i$  is calculated as

$$F_1^i = \frac{2 \cdot P^i(\tilde{Y}_i, Y_i) \cdot R^i(\tilde{Y}_i, Y_i)}{P^i(\tilde{Y}_i, Y_i) + R^i(\tilde{Y}_i, Y_i)} \quad (10)$$

$$P^i(\tilde{Y}_i, Y_i) = \frac{|\tilde{Y}_i \cap Y_i|}{|\tilde{Y}_i|} \quad (11)$$

$$R^i(\tilde{Y}_i, Y_i) = \frac{|\tilde{Y}_i \cap Y_i|}{|Y_i|} \quad (12)$$

Our best-scoring solution submitted to the contest achieved an F1-Score of 0.6561 and a rank of 49 amongst 91 submissions. After the contest, we excluded the HateXplain dataset from the token-classification training process to make the model more suited for the contest dataset and masked out the padding tokens while calculating  $\min_i(\tilde{a}_i)^2$  in Equation 9. We also developed a baseline model

which does not use any additional datasets by training the pre-trained base BERT model with the token-level labeled dataset described in Section 3.1.1. The training hyper-parameters were the same as our other approaches. The F1 scores for the baseline model and the two approaches discussed in Sections 3.2 and 3.3, namely the *sequential adaptation* and the *weakly supervised learning* approaches have been plotted for various thresholds in Figure 3.

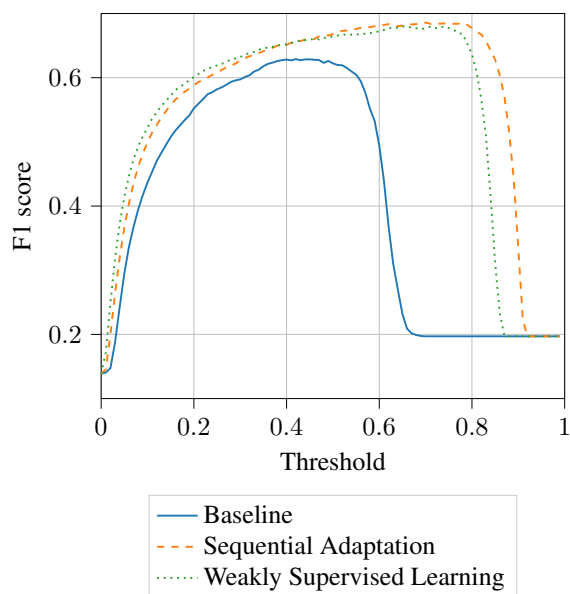


Figure 3: F1 score plotted against threshold

The F1 score at threshold value 0.5 and the maximum F1 score for each approach have been mentioned in Table 1.

Approach	$F1_{0.5}$	$F1_{max}$
Baseline model	0.6224	0.6289
Weakly Supervised	0.6644	0.6799
Sequential adaptation	0.6678	0.6861

Table 1: F1 score metrics for various approaches

Using sentence-level labeled data to incorporate domain-specific knowledge improves the results by a considerable margin. The increase in performance over the baseline model is more profound for the higher threshold value. This can be attributed to the comparatively higher toxicity of the sentence-level labeled datasets when compared with the contest dataset, which makes the models more expert on detecting highly toxic spans. We observe that the sequential adaptation approach outperforms the weakly supervised learning approach. However the latter might be more suitable for large-

scale datasets due to less training time: the sentence classification model has to be trained on the entire sentence-labeled dataset in which the majority of the samples are non-toxic, whereas the model for generating token labels is trained only on toxic samples. In our experiments, the former took 4.25 hours for a single epoch of training and the latter took 1.5 hours for five epochs of training. Even though training the token classification model takes more time in the weakly supervised approach due to larger volume of data samples, the training time difference was only 48 minutes. All the models were trained on the Kaggle Notebooks<sup>4</sup> platform using GPUs.

The model for generating token labels (described in Section 3.3) did not perform as we expected. Out of the 52,640 toxic samples, only 7,629 samples were given a toxicity score of more than 0.5, thus greatly reducing the size of the additional dataset for token classification. However it was successful in correctly identifying the toxic spans. For example, the highlighted parts were given a toxicity score of more than 0.9 in the following sample:

admins **suck!!** the **fucking** admins **suck ass!** i **fucking hate** the people who delete my **fucking shit!!!!!!!**<sup>5</sup>

## 5 Conclusion and Future Work

In this paper, we proposed two solutions to improve the baseline transformer model fine-tuning for the span toxicity detection task. In the first approach, we performed sequential fine-tuning with an additional fine-tuning of the sentence classifier with supplemental public data. In the second approach, we augmented the labeled token dataset with weak supervision and then performed the fine-tuning token classification on the augmented dataset. Our experimental results show that the first approach improves the baseline fine-tuning results by a 0.0572 F1 score and the second approach improves the baseline results by a 0.051 F1 score. An interesting future direction is to improve the weak supervision technique - possibly using other objective functions for relating token labels and sentence labels - and multi-task learning.

<sup>4</sup><https://www.kaggle.com/code>

<sup>5</sup>This sample is from the Jigsaw toxic comment classification challenge dataset (id 13913f443da71ac6) and was labelled as *toxic* and *insult*.

## Acknowledgements

We would like to thank the task organizers for preparing the dataset and organizing the competition, Jennifer Golbeck and her team at the University of Maryland, College Park for providing access to the online harassment dataset, and the anonymous reviewers for their valuable feedback.

## References

- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media, ICWSM '17*, pages 512–515.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjittert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. 2017. [A large labeled corpus for online harassment research](#). In *Proceedings of the 2017 ACM on Web Science Conference, WebSci '17*, page 229–233, New York, NY, USA. Association for Computing Machinery.
- Giannis Karamanolakis, Daniel Hsu, and Luis Gravano. 2019. [Weakly supervised attention networks for fine-grained opinion mining and public health](#). In *Proceedings of the 5th Workshop on Noisy User-generated Text, W-NUT@EMNLP 2019, Hong Kong, China, November 4, 2019*, pages 1–10. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *arXiv preprint arXiv:2012.10289*.
- Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2020. Ethos: an online hate speech detection dataset. *ArXiv*, abs/2006.08328.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. [Toxicity detection: Does context really matter?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.
- Jason Phang, Thibault F evry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.
- Jason Phang, Thibault F evry, and Samuel R. Bowman. 2019. [Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks](#).
- Marek Rei and Anders S oggaard. 2018. [Zero-shot sequence labeling: Transferring knowledge from sentences to tokens](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 293–302, New Orleans, Louisiana. Association for Computational Linguistics.
- Zhi-Hua Zhou. 2018. A brief introduction to weakly supervised learning. *National Science Review*, 5:44–53.



# BennettNLP at SemEval-2021 Task 5: Toxic Spans Detection using Stacked Embedding Powered Toxic Entity Recognizer

Harsh Kataria, Ambuje Gupta, Vipul Mishra\*

Department of Computer Science Engineering

Bennett University, Greater Noida, India

(hkataria99, ambujegupta99)@gmail.com

vipul.mishra@bennett.edu.in

## Abstract

With the rapid growth in technology, social media activity has seen a boom across all age groups. It is humanly impossible to check all the tweets, comments and status manually whether they follow proper community guidelines. A lot of toxicity is regularly posted on these social media platforms. This research aims to find toxic words in a sentence so that a healthy social community is built across the globe and the users receive censored content with specific warnings and facts. To solve this challenging problem, authors have combined concepts of Linked List for pre-processing and then used the idea of stacked embeddings like BERT Embeddings, Flair Embeddings and Word2Vec on the flairNLP framework to get the desired results. F1 metric was used to evaluate the model. The authors were able to produce a 0.74 F1 score on their test set.

## 1 Introduction

Modernization has awarded us with the technology capable of communicating with masses across huge distances in an instant by the touch of a single finger in our palms, the smartphone. With all this innovation every day, it is now more accessible than ever, even in the most rural parts of the world and at very reasonable costs. It has enabled a vast population to share their thoughts and views on popular topics in public forums. People can express themselves in the most creative ways and talk to each other about movies, research, politics, the economy and much more. Some of the key forums and platforms for such activities are Facebook, Twitter, YouTube, Reddit. They allow users to participate in various discussions, discussions that at times may not be very decent. This creates an issue for these platforms as they usually have some of the other policies against indecent content posted by users. On average there are about

350,000 tweets, 510,000 comments, 293,000 status updates on Facebook and Twitter in every 60 seconds (Sayce, 2020). It is humanly impossible for these platforms to check each and everything posted by the users for hate or toxicity. They require an automated method to flag such content.

The motivation for this research task is to achieve some degree of automatic moderation in the social web. It is crucial to moderate social media sites such as Facebook, Twitter and Reddit to be healthy and inclusive. This includes filtering and censoring toxic and hateful content posted online on these public forums. There are automated hate detection NLP models like (Zhang and Luo, 2018) capable of identifying toxic content with acceptable performance. However, they do not identify the specific spans of text that are toxic. This task tries to identify these spans that can be used further to provide insights into a generic text toxicity score. More about the study that this paper is based on can be found from the task organizers paper (Pavlopoulos et al., 2021).

This paper proposes linked lists for data pre-processing and a stacked embedding approach to training this automated system.

The authors' experiment and code for the models ready to be reproduced can be found using the authors' [Github](#) repository.

This paper is organized as follows. It starts with a short abstract describing the paper at a very high level. Then the first section introduces the problem at hand and the task to accomplish. It mentions the authors' approach and the link to their code and models. The second section talks about the background research performed for the task and explains the existing solutions and how this paper is different from them. The third section gives an in-depth understanding of the system approach to solving the tasks where it talks about the embeddings and the stacking method. The fourth section

tells us about the dataset, how it was presented, what it consisted of, the data processing applied to it, the experimental setup and the evaluation metrics. The fifth section is about the methodology, followed by the results section. The seventh section concludes the system description. Finally, we get the references.

## 2 Background research and related works

This problem tackles the challenge of accurately identifying the parts of a toxic text and contributing to making the complete text toxic. There, however, exist systems that can score the toxicity of a full text, sentence or comment like (Zhang and Luo, 2018), but they don't identify the exact part of that sentence that is toxic. There also are systems that can accurately say if a statement is toxic or not and even go on to the extent of identifying the emotion projected by it, for example, if it is positive, negative, humorous, sarcastic, offensive or funny and even the level of these emotions as seen in (Gupta et al., 2020).

A widely known methodology to identify parts of a text is Named Entity Recognition (NER) (Yamada et al., 2020) and its types like the speech (POS) tagging method. NER is a sequence labelling task that recognises entities as per the types of entities it is trained on. These are usually nouns or particular words like names of people or places or organisations and values like monetary numbers and currencies. The sequence labelling task considers a text sequence in which part of it needs to be tagged differently according to the embeddings. Long short term memory, LSTMs (Hochreiter and Schmidhuber, 1997) are usually used for sequence labelling. A variant of LSTMs, the bi-directional recurrent neural network-based BiLSTM, has proved to be very successful at performing accurately at such tasks. It is also seen to be combined with the conditional random field(CRF) decoding layer as seen in (Ma and Hovy, 2016) to get better results.

Open-source tool, Spacy (Honnibal et al., 2020) is a leading tool to create effective and lightweight NER models for datasets with such challenges.

## 3 System Overview

For this task, the authors have used the flairNLP platform (Akbik et al., 2018a) to stack BERT and flair embeddings and create a Named Entity Recognition (NER) model. They have morphed the origi-

nal aim of the NER type tasks to train a model capable of identifying toxicity in a text that is already classified as toxic as a whole. This was possible because, finally, we were using the different embeddings related to toxic parts of the text. These were the embeddings that were trained, which made the task very similar to a NER task. The model created also had a conditional random field layer. Its purpose was to better understand the context around the text's selected parts by considering the neighbouring words.

For this task of sequence labelling, we have stacked embedding by concatenating BERT and flair embedding. This helps in getting a better semantic context out of the concatenated embedding word vectors. Following is a brief about flair and BERT embeddings.

### 3.1 Flair Embedding

Flair embedding (Akbik et al., 2018b) are a type of contextual embedding in which sentences are passed as character into a character level language model to generate word embedding. Contextual string embedding are generated through LSTM because of its ability to keep long term dependency within their hidden state. In Figure 1

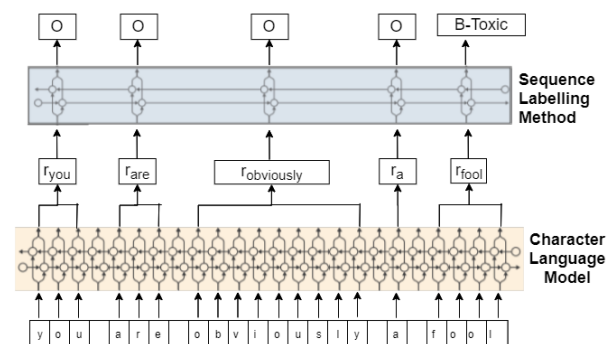


Figure 1: A word is passed as characters to generate flair embedding. After that a CRF layer is used to convert this into a NER problem.

We can see each character of the text "you are obviously a fool" is passed through a bidirectional character-level neural language model, and each word is retrieved. Then it is given to the CRF layer(sequence labelling). Following is a brief explanation of the mathematics behind generating these embedding. In LSTM, the hidden state  $h_t$  represents the past sequence of the character. Both forward and backward language model is used to create these embedding. The mathematical equation of the hidden layers of the forward and back-

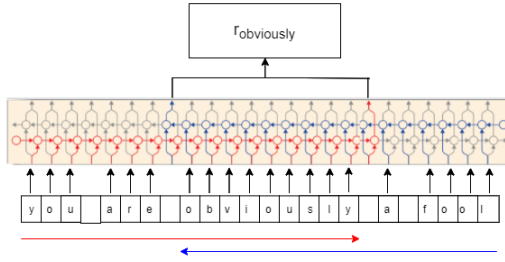


Figure 2: Generating Contextual String Embeddings for the word "obviously". In the forward language model (represented in red), the output hidden state is extracted after the last character in that word. For the backward Language Model (represented in blue), the output hidden state is extracted before the word's first letter. These two output layers are concatenated to form the final embedding

ward model can be seen in equation 1 and 2 respectively.

$$h_t^f = f_h^f(x_{t-1}, h_{t-1}^f, c_{t-1}^f, \theta) \quad (1)$$

$$h_t^b = f_h^b(x_{t+1}, h_{t+1}^b, c_{t+1}^b, \theta) \quad (2)$$

In the above equation,  $f$  and  $b$  are the forward and backward model's notations. Memory cell and parameters of the model are represented by  $c_t$  and  $\theta$ , respectively. The output of the hidden state from both the forward and backward language model are concatenated. In the forward language model, the hidden state output is extracted after the word's last character. Subsequently, for the backward language model, the hidden state's output is extracted before the first character of the word. Both the language model captures the semantic information and then are concatenated to generate the word embedding for that particular word. Let us suppose individual word string begin with  $t_1 \dots t_n$  then the contextual word embedding can be seen in equation 3

$$w_i^{charLM} = \left\{ \begin{array}{c} h_{t_{i+1}-1}^f \\ h_{t_i-1}^b \end{array} \right\} \quad (3)$$

To better understand the concept, figure 2 explains the process by taking an example, and the forward language model is shown in red colour. The backward language model is shown in blue colour. To have a complete understanding of how the contextual word embedding work, one can refer to the (Akbik et al., 2018b) paper.

### 3.2 BERT Embedding

Unlike Flair embedding, BERT embedding (Devlin et al., 2019) are word-level embedding, and it

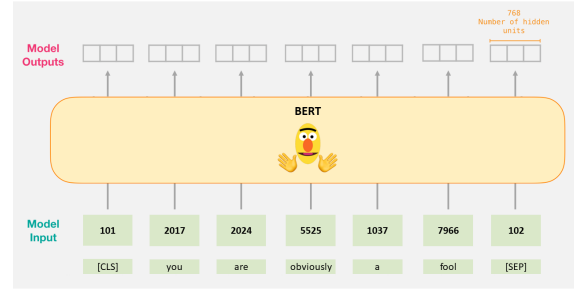


Figure 3: Example Illustrating the process of generating BERT Embedding.

only contains the encoder part of the transformers (Alammar, 2018). In this paper, the authors have used 2 BERT models, i.e. BERT-base, which contains 12 layers, produces an output of 768 units and BERT-Large, which contains 24 layers, produces an output of 1024 units. Generating word embedding can also be classified as feature extraction in which embedding are generated and are fed into the neural network. For the BERT-base model, each layer produces 768 units for every word. To generate BERT embedding, authors have concatenated the last 4 hidden layers. BERT paper (Devlin et al., 2019) also shows that concatenating the last 4 layers yield the best results. BERT has around 30,000 words vocabulary. If the word is not in the vocabulary, then the BERT tokenizer converts it into a sub-word or characters. For example - Word "un-recognized" will be represented as ['un', 're', 'co', 'gni', 'zed']. The first token of a sentence is always ([CLS]), which indicates the sentence's starting. Two separate sentences are separated with ([SEP]) tokens. An example of how the BERT model works is shown in figure 3 which is adopted from (Alammar, 2019) where authors have used "you are obviously a fool" as an example sentence. We can see first the sentence is converted into the sentence convention used by BERT, i.e. adding the required tokens. After adding the tokens, BERT tokenization is used, i.e. words are converted into numbers by referring to the BERT dictionary. BERT tokenization also converts any words in sub-words or character if the required word is not present in the vocabulary file. The ids are then fed into the BERT model, and desired output, i.e. BERT embedding, are generated. Each layer produces 768 units for a single word. The last four (4) hidden layers have been concatenated to get the word embedding in the paper. Now we can put these words embedding to our model to generate results. Figure 4 shows

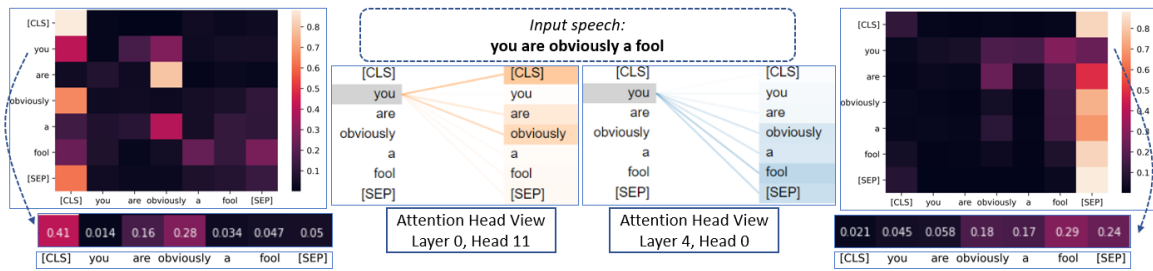


Figure 4: Visualization of Attention in BERT-Base model. Lighter points in the heatmap represents more attention value

the visualization of attention using BERT. The heatmap shows how BERT correlates one word with other.

## 4 Experimental Setup

This section presents a brief overview of the data, the pre-processing it went through and challenges faced in doing so, information about the experiment environment and the evaluation metrics used.

### 4.1 Data

The data was provided to the authors by the task organizers who had acquired it from the Civil Comments Dataset (Borkan et al., 2019). The task organizers then filtered all the text level toxic-labelled text that had been labelled toxic by more than half the annotators, which was around 30,000 in number from a total of 1.2 million posts. Out of these 30,000 text posts, random 10,000 posts were selected and given to crowd annotators to mark the toxic spans from the text. More information about this can be found in the task description paper (Pavlopoulos et al., 2021). The authors finally received a CSV file with two columns with headers as 'spans' and 'text'. Spans column contained a list of character level indices of the toxic entities. Next to it was the complete text that was found to be toxic. For some texts, there existed a corresponding list of empty spans if no toxic span was annotated. Authors randomly collected 558 data points from the dataset (CSV) as a testing set and were left with 5339 data points as the training set.

### 4.2 Linked-list based Pre-processing

The pre-processing stage involved coming up with a method to map the spans before and after cleaning the text. The data provided had a column full of rows with toxic text collected from social media and hence it was naturally in need of cleaning as it contained a lot of abbreviations, punctuation, some

foreign characters, numbers and special characters that were not supposed to be present there as they would create ambiguity to the model. Removing these would bring some uniformity to the model input. In this approach authors faced majorly two (2) challenges.

1. Removing unwanted characters from the original text will produce a cleaner text and that text will be shorter in length as compared to the original text. This will create discrepancy from the spans column, as the spans are given was according to the length of the original sentence.
2. While pre-processing it was important to maintain the sequence of the words in a sentence. For example "You are an Idiot", we have to make sure that after the first word "You" the second word is "are" only.

To tackle the first problem we replaced unwanted(punctuation, numerals etc.) characters with whitespace character so that the length of the sentence remains the same.

**Text :** You.... are, idiot !!

**Pre-processed text :** You are idiot

**Ground spans =** [12,13,14,15,16] = ['idiot']

In the above text, we can see that the unwanted characters are replaced by whitespace character which solves our first problem.

For the second problem, we implemented a linked list data structure. After tokenizing the sentence on whitespace we stored individual word in a single node. The head of the node is attached to the next node which helps us to maintain the sequence

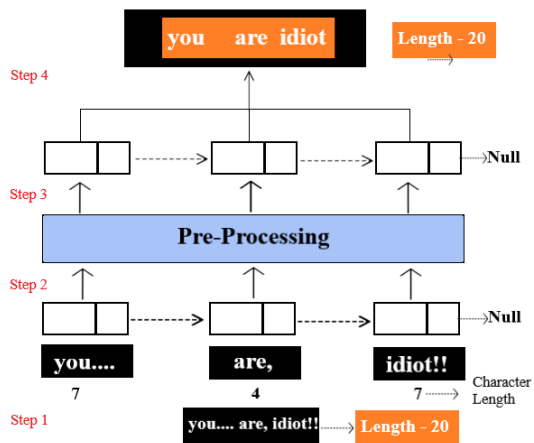


Figure 5: Implementation of linkedlist data structure for pre-processing

of words in a sentence, hence solving our second problem.

This is described stepwise in figure 5

**Step 1:** The original sentence is tokenized on "whitespace". In the example, the length of the original text is 20.

**Step 2:** After tokenization, each word is stored in a node of a linked list data structure and each word from each node is pre-processed i.e. unwanted characters are removed.

**Step 3:** To maintain the sequence of words another linked list is made but in this linked list each word is cleaned. The unwanted characters are replaced with whitespaces in the pre-processing block.

**Step 4:** Words are joined back to form a proper sentence, and the length of the new sentence is 20 as it was of the original sentence.

After getting the pre-processed text, it is ready to be fed into the model.

### 4.3 Experiment Environment

The experimental environment set up by the authors included the use of Python, mostly for scripting along with some of the well known and commonly used python libraries like NumPy, pandas, flair(flairNLP), re(regex). They used Jupiter notebook for python ide along with python version 3.7. Google Colab (GPU and TPU) was also used for training the models.

### 4.4 Evaluation metrics

Authors have used F1 score (Da San Martino et al., 2019), precision and recall (Goutte and Gaussier, 2005) in order to evaluate the performance of the models. The task organisers also used F1 score to evaluate and rank the challenge responses. Equation 4 represents the mathematical formula of calculating the F1 score.

$$F_1^l = \frac{2 \cdot P^l(X_i, Y) \cdot R^l(X_i, Y)}{P^l(X_i, Y) + R^l(X_i, Y)} \quad (4)$$

In equation 4,  $F_1^l$  represents F1 score of the system  $i$ , which is calculated for the text  $l$ . The predicted values are represented by  $X_i$  whereas the ground truth is represented by  $Y$ .  $P$  and  $R$  represents Precision and Recall values with their mathematical calculations shown in equations 5 and 6 respectively where  $S$  represents the Set function.

$$P^l(X_i, Y) = \frac{|S_{X_i}^t \cap S_Y^t|}{|S_{X_i}^t|} \quad (5)$$

$$R^l(X_i, Y) = \frac{|S_{X_i}^t \cap S_Y^t|}{|S_Y^t|} \quad (6)$$

Here,  $|\cdot|$  represents Cardinality which can be interpreted as the length of the finite set.

The overall F1, precision and recall of the models' performance was obtained by calculating the mean of individual scores of every text in our test set of 558 data points.

## 5 Methodology

In this paper authors have used the concept of stacked embeddings i.e. to generate a particular embedding for a word, flair gives you the flexibility to concatenate different word embeddings together to get better results. Equation 7 depicts concatenating flair and GloVe embedding.

$$we_i = \left\{ \begin{array}{l} we_i^{Flair} \\ we_i^{GloVe} \end{array} \right\} \quad (7)$$

Here word embedding is denoted by  $we$ . In this paper, the result of four (4) models have been displayed. Table 1 shows different parameters used by authors in different models. The concept of early stopping and adaptive learning rate are used to generate these results. The learning rate would be halved if the model does not show improvement consecutively 4 times in a row. In this case, the training automatically stops when the learning rate becomes too small for example  $LR=6.2500e-05$ .

Parameters	Model 1	Model 2	Model 3	Model 4
Epochs	30	39	30	36
Learning Rate	0.001	0.001	0.001	0.001
Mini Batch Size	8	8	8	8
Embeddings	Flair,BERT-large-uncased, CharacterEm-beddings	Flair,BERT-large-uncased, CharacterEm-beddings,GloVe	Flair,BERT-base-uncased, CharacterEm-beddings,GloVe	Flair,BERT-base-uncased, CharacterEm-beddings

Table 1: Parameters of different Proposed Models

## 6 Result

The parameter details of the 4 different models and the embeddings used are listed in table 1. It lists the number of epochs the model was trained on with the initial learning rates and batch sizes. Model 2 ran for the most epochs before being auto-stopped as no improvements were seen in the last 4 epochs and the learning rate parameter got too small due to the adaptive learning rate function. All the models had Flair and Character embeddings with the variation of GloVe and BERT-uncased embeddings. Table 2 lists different models used

Model	F1	Precision	Recall
Model 1	0.748	0.971	0.929
Model 2	0.737	0.967	0.925
Model 3	0.726	0.968	0.916
Model 4	0.724	0.973	0.909

Table 2: Experimental results

and their respective F1-scores, precision and recall values for our test set. It can be inferred that model 1 i.e. stacked embeddings with Flair, BERT-large-uncased, CharacterEmbedding performed the best with an F1 score of 0.748 with precision and recall of 0.971 and 0.929 respectively. It was able to predict toxic spans closest to the ground truth. The other models are not too behind than this but it seems that model 4 did not have too much of a difference when stacked with GloVe embeddings as seen in model 3.

Figure 6 presents the comparative analysis of F1 scores achieved by the 4 models proposed by the authors in table 1 and the best performing model from the NLRG system (Chhablani et al., 2021) and the UniParma system (Karimi et al., 2021). The authors of NLRG have used a BERT based RoBERTa token classification method to reach their best F1 score of 0.689. The authors of

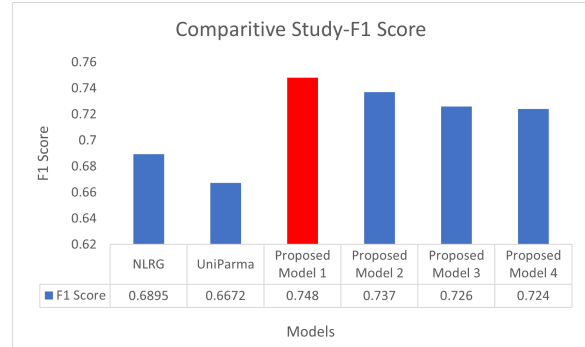


Figure 6: Comparative Analysis of F1 scores

UniParma have used CharacterBERT and the bag of words(BOW) method to get their F1 score of 0.66. Proposed model 1 (highlighted in red) was the best performing model with a 0.748 F1 score.

**Text :** You.... are, idiot !!

**Pre-processed text :** You are idiot

**Ground spans**=[12,13,14,15,16]=['idiot']

**Predicted spans**=[12,13,14,15,16]=['idiot']

The above example displays the input text, the pre-processed clean text with the toxic word "idiot" highlighted in red. Below them is the ground truth for the spans of this toxic word and then there are the correct predicted spans for it.

## 7 Conclusion

We support the systematic development for identifying toxic spans. We have successfully been able to deploy a linked list approach to prepare the data and then train it using the stacked embeddings method and produce empirical results. This task can prove to be useful in providing a better analysis in the censoring of toxic posts on the internet.

## References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018a. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018b. Contextual string embeddings for sequence labeling.
- Jay Alammar. 2018. [The illustrated transformer](#).
- Jay Alammar. 2019. [A visual guide to using BERT for the first time](#).
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Gunjan Chhablani, Yash Bhartia, Abheesht Sharma, Harshit Pandey, and Shan Suthaharan. 2021. [Nlrg at semeval-2021 task 5: Toxic spans detection leveraging BERT-based token classification and span prediction techniques](#).
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeno, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5640–5650.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#).
- Cyril Goutte and Eric Gaussier. 2005. [A probabilistic interpretation of precision, recall and f-score, with implication for evaluation](#). In *Proceedings of the 27th European Conference on Advances in Information Retrieval Research, ECIR'05*, page 345–359, Berlin, Heidelberg. Springer-Verlag.
- Ambuje Gupta, Harsh Kataria, Souvik Mishra, Tapas Badal, and Vipul Mishra. 2020. [Bennettnlp at semeval-2020 task 8: Multimodal sentiment classification using hybrid hierarchical classifier](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation, SemEval@COLING 2020, Barcelona (online), December 12-13, 2020*, pages 1085–1093. International Committee for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. [Uniparma @ semeval 2021 task 5: Toxic spans detection using characterBERT and bag-of-words model](#).
- Xuezhe Ma and Eduard Hovy. 2016. [End-to-end sequence labeling via bi-directional lstm-cnns-crf](#). pages 1064–1074.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- David Sayce. 2020. [The number of tweets per day in 2020](#).
- Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. [Luke: deep contextualized entity representations with entity-aware self-attention](#). *arXiv preprint arXiv:2010.01057*.
- Ziqi Zhang and Lei Luo. 2018. [Hate speech detection: A solved problem? the challenging case of long tail on twitter](#). *Semantic Web*, Accepted.

# UoT-UWF-PartAI at SemEval-2021 Task 5: Self Attention Based Bi-GRU with Multi-Embedding Representation for Toxicity Highlighter

**Hamed Babaei Giglou**

Department of Computer Science  
University of Tabriz, Tabriz, Iran  
h.babaei98@ms.tabrizu.ac.ir

**Taher Rahgooy**

Department of Computer Science  
University of West Florida, Florida, USA  
trahgooy@students.uwf.edu

**Mostafa Rahgouy**

Part AI Research Center  
Tehran, Iran  
mostafa.rahgouy@partdp.ai

**Jafar Razmara**

Department of Computer Science  
University of Tabriz, Tabriz, Iran  
razmara@tabrizu.ac.ir

## Abstract

Toxic Spans Detection(TSD) task is defined as highlighting spans that make a text toxic. Many works have been done to classify a given comment or document as toxic or non-toxic. However, none of those proposed models work at the token level. In this paper, we propose a self-attention-based bidirectional gated recurrent unit(BiGRU) with a multi-embedding representation of the tokens. Our proposed model enriches the representation by a combination of GPT-2, GloVe, and RoBERTa embeddings, which led to promising results. Experimental results show that our proposed approach is very effective in detecting span tokens.

## 1 Introduction

With the massive increase in social interactions on online social networks, keeping discussions fruitful is a central concern for platform providers. Indeed, abusive (e.g., bullying, profanity, hate speech), damaging the reputation of a platform. Thus, it is necessary to be detected by automated machine learning systems because of huge amount of data. However, the previous works mostly focus on whether the given document(e.g. comment) is toxic or not. Detecting the span tokens in the document may be more beneficial. For example, it can prevent users to use span tokens before they send their post. Also, it is useful to filter span tokens before learning AI chatterbots instead of removing the whole documents.

In this paper, we propose a self-attention-based bidirectional gated recurrent unit (BiGRU) with a multi-embedding representation of the tokens. Our proposed model enriches representation showed very promising results.

The rest of the paper is organized as follows. Section 2 provides background and presents some related works on TSD in general. Section 3 intro-

duces our BiGRUs model. Results are covered in Section 4. In Section 5 we draw a conclusion.

## 2 Related Research

A toxic post (comments) is defined as a rude, disrespectful, or unreasonable comment that is likely to make other users leave a discussion. A goal of toxic comment classification is to give a right to freedom of expression on the web.

Training complex neural networks (NN) requires enough datasets of toxic comments. Word embeddings are the basis of the NNs when working with text data. NNs for toxic comment classification use Recurrent Neural Networks (RNN) layers such as Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) (Chung et al., 2014) layers. Similarly, the attention mechanism for NNs has been successfully applied to toxic comment classification (Pavlopoulos et al., 2017; van Aken et al., 2018). In semi-automated content-moderation, attention can be considered as a highlighter of abusive or toxic words.

Badjatiya et al. (2017) proposed a system for a hate-speech task based on deep-learning with a combination of LSTM, random embedding, and gradient boosted decision trees as the best model. They used random embedding, GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2016) representation for experimentation. They concluded that a combination of CNN and LSTM with FastText or GloVe embedding as features for gradient boosted decision trees can not yield better results.

van Aken et al. (2018) held an in-depth error analysis, and based on their comparance with different deep learning and shallow approaches; they observed three common challenges: out-of-vocabulary words, long-range dependencies, and



multi-word phrases. They experimented with various deep learning models, including CNN, LSTM, BiLSTM, GRU, BiGRU, and Attention mechanism with GloVe and FastText embeddings to tackle these challenges. Their experimentation on two datasets showed that BiGRU with Attention mechanism with GloVe and FastText representations achieved promising results with respect to other models. However, in the final, they proposed an ensemble approach that outperforms all individual models.

In (Pavlopoulos et al., 2017) they proposed a deep classification-specific attention mechanism with BiGRU to highlight suspicious words for automatic and semi-automatic content moderations.

### 3 Proposed Method

In this section, we describe the details of our proposed neural network model. Our proposed approach aims to predict whatever the tokens of given comments are toxic or not. Figure 1 depicts an overview of our proposed method.

At first, in a dataset, original posts are tokenized and preprocessed. Each token at each post is labeled as a toxic or not toxic token by their spans. Then, a multi-embedding representation of tokens is created. Next, the Bidirectional GRUs (BiGRUs) models are applied to extract the higher-level feature sequences with sequential information from multi-embeddings. After that, a self-attention mechanism computes attention weight between each pair of elements in a single sequence. Finally, the generated output feature sequences from self-attention-based BiGRUs are fed into the fully-connected dense layer and then into the final prediction module to determine the prediction. In the following, we describe each component elaborately.

#### 3.1 Preprocessing

First, each comment is tokenized into words with their spans. Next, tokens are preprocessed; the preprocessing consists of lowercasing and removing punctuations, special characters, numbers, Unicodes, smileys, and emojis. After preprocessing, empty tokens are removed (tokens which empty spaces). Finally, for a post, toxic or not-toxic labels based on grand truth assigned to preprocessed tokens. In total, we obtained **21790** unique words. The obtained words are used as a vocabulary. In the next step, a multi-embedding is used to extract

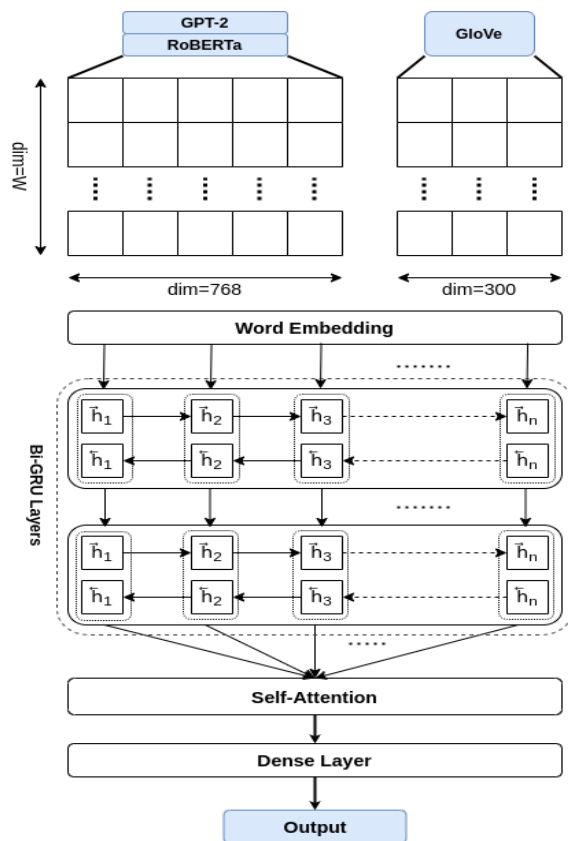


Figure 1: Architecture of Proposed Model

features for unique words to create an embedding matrix for modeling.

#### 3.2 Multi-Embedding Layer

Toxic comments often use obfuscations, for example, "f\*k u", "Son of a B\*\*\*\*", "\*\*\*\*k them.". Also, misspelled and abbreviation words are common in online discussions. Word2Vec (Mikolov et al., 2013) and GloVe fail to find a good representation of these words because words never occurred in training time. These words are out-of-vocabulary (OOV) (Risch and Krestel, 2020). However, we can take advantage of failing representations to represent toxic tokens that are not in vocabulary by setting their representations to zero. Regarding this hypothesis, if GloVe or Word2Vec contains OOV, we can set their token representations to zero and use language model embeddings to find subset word representation. By concatenating these two types of representation for each token in the sequence, we can build awareness representation of sequences.

For this purpose, we used the multi-embedding representation of tokens.

**GloVe** is a global log-bilinear regression model

with a weighted least-square objective that combines the advantages of global matrix factorization and local context windows. It leverages statistical information by training only on the nonzero elements in a word-word co-occurrence matrix in a large corpus.

**GPT-2** (Radford et al., 2019) is an unsupervised transformer language model for general-purpose learners. It is trained on WebText, which contains over 8 million documents.

**RoBERTa** (Liu et al., 2019) is an optimized version of BERT (Devlin et al., 2018) model. It builds on BERT’s language masking strategy. RoBERTa modifies key hyperparameters in BERT, including removing BERT’s next sentence pretraining objective and training with much larger mini-batches and learning rates.

To empower representation, we combined RoBERTa and GPT-2 representations (by summing) and then concatenated them with GloVe (840B tokens, 2.2 vocab). We achieved representation matrix with  $W \times 1068$  dimension for training vocabs, where  $W$  is the number of vocabs in training. During analysis, we found nearly  $6k$  OOV words in training, which GloVe does not produce a representation for them. It is a significant number regarding the training vocabulary size.

### 3.3 Bidirectional GRUs

Bidirectional Gated Recurrent Unit (BiGRU) in the central part of Figure 1 is a bidirectional version of GRU. The GRU allows to adaptively capture dependencies from large sequences of data without discarding information from earlier parts of the sequence. BiGRU combines the forward hidden layer with the backward hidden layer, which can process each sequence in both left-to-right and right-to-left order to embed the sequential dependencies in both directions. The first BiGRU layer is used to process each sequence token-by-token and produce an intermediate representation. Then, this intermediate representation is used as input for the second BiGRU layer.

### 3.4 Self-Attention Layer

The words in sequences sometimes are related to each other, like "Son", "of", "B\*\*\*" and sometimes are not related. To determine how two tokens are related, Self-Attention Networks (SANs) (Luong et al., 2015) produce the output with the same size as input sequences by considering the attention

of all input tokens with each other. It learns the important interactions between tokens.

### 3.5 Dense Layer

The dense layer or feed-forward layer is the most general-purpose deep learning layer. The dense layer consists of 50 neurons for the weighted linear combination of inputs with the activation function of tanh to squashes the input to the range  $[0, 1]$ .

### 3.6 Prediction Module

The final layer of the network has three neurons, and its returned value is a continuous numerical value. We used the sigmoid activation function to produce a probability vector. For loss function and optimization, we employ Sparse Categorical Cross entropy and RMSprop, respectively.

## 4 Results

### 4.1 Dataset

For toxic span detection tasks (Pavlopoulos et al., 2021) posts from publicly available Civil Comment dataset are used for annotations of particular toxic spans in toxic comments. The task consists of 7939 annotated comments with their toxic spans for training and 2000 for the test. However, we treat 690 samples of trial data as a development set for our investigations.

### 4.2 Evaluation

For evaluation of participating systems in the challenge, F1 score presented in (Da San Martino et al., 2019) was used. If we consider  $A_i$  to return a set  $S_{A_i}^t$  of character offsets for the part of the post found to be toxic, and similarly  $G^t$  be the character offset of the grand truth annotation of  $t$ . We can compute F1 score of system  $A_i$  with respect to the  $G$  for post  $t$  as follows:

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{S_{A_i}^t}$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cup S_G^t|}{S_{A_i}^t}$$

If  $S_G^t$  and  $S_{A_i}^t$  are empty for some post  $t$ , then  $F1 = 1$ , and otherwise  $F1 = 0$ . In final, to obtain a single score for system  $A_i$ , the  $F1$ s averaged over all the posts  $t$  of the evaluation dataset.

Method	GloVe	GPT-2	RoBERTa	RG	GoR	GoG	Ensemble
Results on dev set							
BiLSTM	0.619	0.580	0.634	0.647	0.627	0.621	<b>0.655</b>
BiGRU	0.597	0.641	0.621	0.664	0.637	<b>0.668</b>	0.643
BiLSTM + Attention	0.581	0.615	0.620	0.638	0.607	0.445	<b>0.663</b>
BiGRU + Attention	0.572	0.649	0.562	0.521	0.664	0.601	<b>0.668</b>
Results on Test set							
BiLSTM	0.627	0.666	0.663	0.669	0.665	<b>0.680</b>	0.673
BiGRU	0.633	0.623	0.660	0.662	0.648	0.670	<b>0.680</b>
BiLSTM + Attention	0.653	<b>0.676</b>	0.657	0.600	<b>0.668</b>	0.559	0.633
BiGRU + Attention	0.639	0.659	0.644	0.627	0.640	<b>0.678</b>	<b>0.677</b>

Table 1: Experimental Results on Trial (dev) and Test sets. RG refers to the ensemble of RoBERTa and GPT-2 embeddings. Similarly, GoR, refers to the ensemble of GloVe and RoBERTa embeddings, and GoG refers to the ensemble of GloVe and GPT-2 embeddings

### 4.3 Results

For all experimentation, we used Google Colab free GPU<sup>1</sup> to train our models with 10 epochs. We set the batch size to 32, and we pad the comments to the 215 sequence length. We obtained 6330 OOV out of 21790 words in the vocabulary, which GloVe does not produce a representation for them. For experimentation, we used BiLSTM and BiGRU models with SAN followed by a dense layer. Also, we examined representation combinations of GloVe, GPT2, and RoBERTa and reported them in Table 1 for dev (trial) and test sets. According to the experimentations, all models perform well when multi-embedding representation is utilized.

In the first part, we took GloVe representation as our baseline representation. Regarding this representation, in most cases, GPT-2 and RoBERTa perform well (6 cases for GPT-2, and 7 cases for RoBERTa). It shows how much the contextualized representations are useful; however, it is hard to tell among GPT-2 and RoBERTa which one is performing well.

In the second part, we combined different representations that achieved a higher averaged value of F1 score in all cases. Except for one case, namely BiLSTM + Attention, the differences between representation by GPT-2 and GoR is 0.008. In general, an ensemble of embeddings achieved a higher score than single representations.

In the final, because of two reasons, we considered the BiGRU + Attention model with multi-embedding representations as the final model for this task. First, It achieved a higher averaged F1 according to the dev set, and the second higher

averaged F1 score according to the test set. The second reason is that the margin between the BiGRU+Attention model in the dev and test set was less than the others (0.009). Submitted model to the competition achieved averaged F1 score of **0.677** and place **23** of the competition among 91 teams.

We shared the implementation of the proposed model in GitHub<sup>2</sup> for the research community.

## 5 Conclusion

In this paper, we presented our approach for SemEval-2021 Task 5: Toxic Span Detection. We tried to tackle the problem by employing multi-embedding and deep learning techniques. We conducted some experiments using different models. For example, we implemented a BiLSTM model, BiLSTM with SA, BiGRU, and BiGRU with SA, but the model that gave a promising result and relied on BiGRU with SA model.

## References

- Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. [Challenges for toxic comment classification: An in-depth error analysis](#). In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42, Brussels, Belgium. Association for Computational Linguistics.
- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. [Deep learning for hate speech detection in tweets](#). WWW '17 Companion, page 759–760, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

<sup>1</sup><https://colab.research.google.com>

<sup>2</sup><https://github.com/HamedBabaei/semEval2021-task5-tsd>

- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.
- Junyoung Chung, Çağlar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. [Empirical evaluation of gated recurrent neural networks on sequence modeling](#). *CoRR*, abs/1412.3555.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. [Effective approaches to attention-based neural machine translation](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#). *CoRR*, abs/1301.3781.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- Julian Risch and Ralf Krestel. 2020. Toxic comment detection in online discussions. In *Deep Learning-Based Approaches for Sentiment Analysis*, pages 85–109. Springer.

# YoungSheldon at SemEval-2021 Task 5: Fine-tuning Pre-trained Language Models for Toxic Spans Detection using Token classification Objective

Mayukh Sharma, Ilanthenral Kandasamy, W.B. Vasantha

School of Computer Science and Engineering

Vellore Institute of Technology

Vellore, Tamil Nadu, India

04mayukh@gmail.com, ilanthenral.k@vit.ac.in,

vasantha.wb@vit.ac.in

## Abstract

In this paper, we describe our system used for SemEval 2021 Task 5: Toxic Spans Detection. Our proposed system approaches the problem as a token classification task. We trained our model to find toxic words and concatenate their spans to predict the toxic spans within a sentence. We fine-tuned Pre-trained Language Models (PLMs) for identifying the toxic words. For fine-tuning, we stacked the classification layer on top of the PLM features of each word to classify if it is toxic or not. PLMs are pre-trained using different objectives and their performance may differ on downstream tasks. We, therefore, compare the performance of BERT, ELECTRA, RoBERTa, XLM-RoBERTa, T5, XLNet, and MPNet for identifying toxic spans within a sentence. Our best performing system used RoBERTa. It performed well, achieving an F1 score of 0.6841 and secured a rank of 16 on the official leaderboard.

## 1 Introduction

Internet and social networking sites have brought people together by providing a simple yet effective method of communication. Over the years people used it to exchange positive ideas but recently, there has been a rise in toxic content and hate speech over the internet (Zampieri et al., 2019, 2020). Most datasets (Fortuna et al., 2020) dealing with the problem of toxic, offensive, or hateful content aim to classify the entire text belonging to a particular class. They do not identify the parts of the text that make it toxic. Manual filtering of toxic data is tough and can cause mental and emotional stress to annotators (Zampieri et al., 2019). An automatic system with the ability to identify toxic text and highlighting toxic spans can be useful for the moderators. It will help save time and prevent stress caused by reading long texts. SemEval 2021 Task 5: Toxic Spans Detection (Pavlopoulos et al.,

2021) draws attention to the problem of identifying toxic spans present in a sentence.

Our proposed system makes use of a word-level classifier for detecting the offensive words present in a sentence. The offsets of the toxic words can then be concatenated to find the toxic spans. We made use of pre-trained language models (PLMs) for building our classifier. We experimented with BERT (Devlin et al., 2019), ELECTRA (Clark et al., 2020), RoBERTa (Liu et al., 2019b), XLNet (Yang et al., 2020), MPNet (Song et al., 2020), T5 (Raffel et al., 2020), and XLM-RoBERTa (Conneau et al., 2020) to compare their performance on the task of toxic spans detection. Owing to the increase in the number of pre-trained language models choosing the correct model is an important decision as these models contain millions of parameters and are expensive to train. So, we present a comprehensive analysis of the performance of different models, which can serve as a baseline for future work.

Our best performing system was fine-tuned using RoBERTa and attained an F1 score of **0.6841**. It was ranked **16** on the official leader board. We used different PLMs for fine-tuning and found exceedingly small variations in their performance. Further analyzing our model's performance on the test set we observed that it is essential for the model to not only detect toxic spans but also decide if it needs to predict toxic spans for that sample or not. Our code is available online<sup>1</sup> for method replicability.

## 2 Background

Identification of toxic/offensive content is an important task in natural language processing. It is essential for the moderation of harmful content over social media sites that might hurt the sentiments of individuals, groups, or communities at large. Much

<sup>1</sup><https://github.com/04mayukh/YoungSheldon-at-SemEval-2021-Task-5-Toxic-Spans-Detection>

work has been done on the identification of offensive content. OffensEval 19, 20 (Zampieri et al., 2019, 2020) provide a comprehensive analysis of methods useful for the identification of offensive content. SemEval 2020 Task 8: Memotion analysis (Sharma et al., 2020) presented with a dataset of internet memes with one sub-task to detect and quantify offensive content. Work done in (Brassard-Gourdeau and Khoury, 2019) explores different aspects of sentiment detection and their correlation to toxicity. (Pavlopoulos et al., 2020) covers the effect of context on toxicity. (D'Sa et al., 2020) uses BERT and FastText for toxicity detection. (Kurita et al., 2019) covers several attacks to by-pass toxic content filters and methods to make the filters robust to such attacks. Recent state-of-the-art systems (Wiedemann et al., 2020; Wang et al., 2020; Liu et al., 2019a; Nikolov and Radivchev, 2019) performed well in identifying offensive content. Work done in (Gröndahl et al., 2018) shows that although recent systems perform well on given datasets, very slight changes made by adversaries may fool the models. Adding words like “love” to offensive tweets may make it less offensive.

Identifying toxic content is an important NLP task. It is useful in moderating online content over the web having millions of users. Most problems deal with labeling the entire content as toxic/non-toxic. None of the previous work has tried to identify spans within a text that makes it toxic. SemEval-2021 Task 5: Toxic Spans Detection aims to bring attention to this problem via the task defined as: Given a dataset  $D$  of sentences, the objective of the task is to learn a classification function that can predict the toxic spans  $T$  present in the given sentence. The content of the provided dataset  $D$  was in English.

**Dataset statistics:** The dataset for the task consisted of character offsets for toxic spans present for each text sample. The span consisted of single words as well as a collection of words. Table 1 shows the count of samples having different number of toxic words.

From Table 1 we can infer that samples with toxic words within the range of one to three form a major component of the dataset. In the test set, samples with no toxic words were significantly more than the training and development set. Toxic words with the highest frequency of occurrence present in the training set are given in Table 2. We observed that toxic words contained stopwords (the,

a, and, of) which are generally not toxic when used independently. These stopwords can exist as part of multiword toxic spans.

### 3 System Overview

#### 3.1 Pre-trained Language Models

Natural language processing tasks are data intensive. Training deep neural networks for NLP tasks requires large amounts of training data that might not always be available. To overcome this problem researchers proposed pre-training large language models which can be fine-tuned on various downstream tasks. Pre-training involves training general representations of text to understand its syntactic and semantic relations. The main advantage of pre-training is that it can be done on unlabelled text corpus allowing training on a large amount of textual data. The pre-trained language models can then be used across various downstream tasks by fine-tuning them on task-specific datasets.

#### 3.2 Brief overview of used PLMs

**BERT:** It is a bidirectional language model based on the Transformer architecture (Vaswani et al., 2017). It uses Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) as a pre-training objective.

**ELECTRA:** It is one of the most recent models and is inspired by generative adversarial networks. It introduces Replaced Token Detection (RTD) pre-training objective.

**RoBERTa:** It is a modification of BERT proposed by Facebook. It uses dynamic masking as a part of the pre-training objective. NSP was removed and the model was pre-trained on larger data for more time.

**XLNet:** It is a generalized auto-regressive pre-training method using the best of both Auto Regressive (AR) and Auto Encoding (AE) modeling techniques. It makes use of permutation language modeling (PLM) objective for pre-training.

**MPNet:** It was proposed by Microsoft. It overcomes the pre-train fine-tune discrepancy in XLNet. It makes use of both PLM and MLM to map the dependencies among predicted tokens as well as use full positional information in a sentence.

**T5:** It was proposed by Google and aimed to re-frame all NLP tasks into a single text-to-text format where both inputs and outputs are always strings. It used a masking objective similar to BERT and used teacher forcing for pre-training.

No. of Toxic words per sentence (N)	N = 0	N>0 and N<=3	N>3 and N<=7	N>7	Total
Train	486	6216	742	495	7939
Development	43	543	72	32	690
Test	394	1541	53	12	2000

Table 1: Number of samples with different frequencies of toxic words.

Toxic Word	Frequency
stupid	1237
idiot	668
the	581
idiots	428
a	383
and	350
of	288
ignorant	277
stupidity	276

Table 2: Most frequent toxic words.

**XLm-RoBERTa:** It is a multi-lingual model trained by Facebook AI on more than 100 languages. It made use of the Transformer architecture (Vaswani et al., 2017) with multilingual MLM (Devlin et al., 2019; CONNEAU and Lample, 2019) using only monolingual data as a pre-training objective.

### 3.3 Modelling as Token Classification Task

The given dataset provided spans of toxic content in a statement. Each sentence could contain multiple toxic spans. Another important thing to note was that a toxic span could comprise more than one word. We extracted all toxic words using the toxic spans. If a span contains over one word, it was further processed to extract individual words. Once we found all the toxic words, we split the original sentence to label the toxic/non-toxic words. Before splitting the original sentence, we removed extra whitespace and newline characters. We removed any punctuation before or after the word. Punctuations present within the words were not removed. Figure 1 shows an example of the process. The toxic spans have been highlighted in red in the original sentence which, we convert into an array of words labeled as toxic/non-toxic.

The next step is to prepare the data for fine-tuning on pre-trained language models. PLMs use tokenization to break the original words into sub-words. Different models use different tokenization techniques like Byte-Pair-Encoding (BPE)

(Sennrich et al., 2016), WordPiece (Schuster and Nakajima, 2012), and SentencePiece (Kudo and Richardson, 2018). One advantage of using tokenization is that it helps to reduce the vocabulary size. One challenge it poses for token classification tasks is which sub-word to use for classification. Different models also add special tokens like [CLS], [SEP], start, end tokens which are not required for the token classification task. In our approach, we used the first sub-word of the tokenized word for classification. We masked the remaining sub-words and special tokens while computing the loss. The sub-words were masked only during loss computation and not while being passed through the model. This allowed all sub-words to learn dependencies within the sentence. Figure 2 shows the tokenized words and their corresponding labels using the BERT tokenizer.

### 3.4 Fine-tuning

We used a simple approach for fine-tuning the model for token classification. We used a token classifier on top of features learned by PLMs. Our classifier consisted of three layers on top of PLM features. First was the batch normalization layer, followed by a dropout layer. The final layer was a time-distributed dense layer over features of each tokenized word containing a single neuron and a sigmoid activation to predict if the given token is toxic/non-toxic.

### 3.5 Masked Loss

As described, we reduced the problem to a token classification task where we predict the label for each word. We used binary cross-entropy loss for the fine-tuning process. In cases where the original word is broken down into multiple sub-words, we used only the first sub-word for calculating the loss. We created masks for each sentence to store the position of words/sub-words. Cross-entropy loss was calculated for required sub-words/words using the masks and then summed up over all tokens in a sentence. The summed value was the loss for a given sentence.

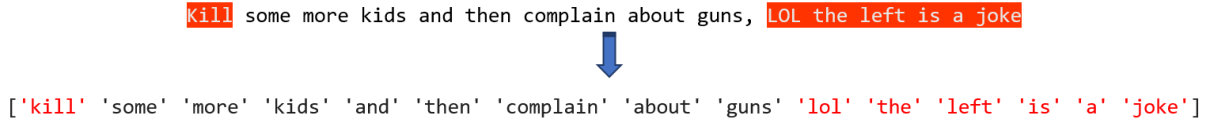


Figure 1: Converting toxic spans to toxic words.

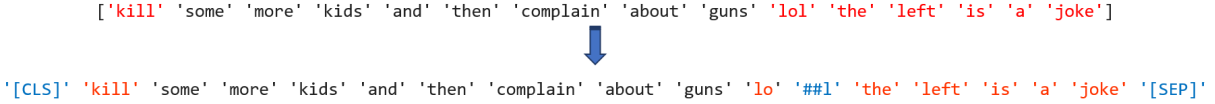


Figure 2: Tokenization for using PLMs. Sub-words(except first sub-word) and special symbols were masked.

## 4 Experimental Setup

### 4.1 Hyperparameters and Training

Our models were developed on Keras<sup>2</sup> (Chollet et al., 2015) using HuggingFace’s<sup>3</sup> implementation of transformer<sup>4</sup> (Wolf et al., 2020) models. We fine-tuned the models on TPU’s on Google Colab. We fixed the sequence length of input to 150 tokens. We padded/truncated the sequences according to their length. Our model was fine-tuned using the AdamW optimizer(Loshchilov and Hutter, 2019) with a linear learning rate decay against masked binary cross-entropy loss. We experimented with learning rates of 1e-4, 3e-5, 4e-5, 5e-5 for each PLM architecture. Fine-tuning was done for 4 epochs. Each PLM architecture with the best performance on the development set was used for making final predictions on the test set.

### 4.2 Predicting Toxic Span Offsets

Our model was trained to find the toxic words. In case the word was tokenized into sub-words, we used the first sub-word to determine the toxic nature of the entire word. We stored flag values for each sentence to find the correct label for each word during prediction. Once we found the toxic words, we searched for them in the original un-processed sentences. We concatenated the spans for all predicted toxic words which was the final expected output.

### 4.3 Evaluation Metric

The performance of the model was evaluated using the F1 score as described in (Da San Martino et al., 2019). Let system  $A_i$  return a set  $S_{A_i}^t$  of character offsets found toxic for post  $t$ . Let  $G_t$  be

Model	F1 Score	
	Dev	Test
BERT-base	0.6654	0.6812
ELECTRA-base	0.6710	0.6804
RoBERTa-base	0.6676	<b>0.6842</b>
XLM-RoBERTa-base	0.6519	0.6775
T5-large	0.6658	0.6811
XLNet-base	0.6714	0.6817
MPNet-base	<b>0.6750</b>	0.6800

Table 3: Model performance on Test set.

ground truth annotation for  $t$ . F1 score of system  $A_i$  with respect to ground truth values  $G$  for post  $t$  is calculated as follows:

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)}$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_{A_i}^t|}$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{|S_G^t|}$$

where  $|\cdot|$  represents the cardinality of the set. If  $S_G^t = 0$  i.e no toxic spans are present in  $t$  then  $F_1^t(A_i, G) = 1$  if  $|S_{A_i}^t| = 0$  else  $F_1^t(A_i, G) = 0$ . Finally,  $F_1^t(A_i, G)$  was averaged over all posts  $t$  present in dataset D to obtain single score for system  $A_i$ .

## 5 Results and analysis

Table 3 shows the performance of our proposed model on different PLMs. Learning rate of 1e-4 was used for ELECTRA, 4e-5 for MPNet, and 5e-5 for the remaining PLMs to obtain the above-mentioned results. RoBERTa had the best performance on the test set while MPNet had the best

<sup>2</sup><https://keras.io>

<sup>3</sup><https://huggingface.co/transformers>

<sup>4</sup><https://huggingface.co>



Model	No. of toxic words = 0		No. of toxic words >0	
	F1 = 1	F1 = 0	F1 = 1	F1 = 0
RoBERTa	24	370	1061	96
BERT	24	370	1050	99
ELECTRA	22	372	1007	76
MPNet	18	376	1063	101
T5	24	370	1041	92
XLNet	32	362	1059	112
XLM-RoBERTa	19	375	1056	104

Table 4: Performance analysis on test samples containing no toxic words vs containing one or more toxic words.

Words	Frequency
stupid	55
ignorant	32
idiot	27
garbage	16
fool	13
pathetic	13
moron, ass, white, dumb, stupidity, idiots	12
racist	10
trash, crap	9

Table 5: Words predicted as toxic in Test samples containing no toxic spans.

performance on the development set. Our best performing model achieved a best F1 score of 0.6842 on the test set and was ranked 16 on the official leader board.

We further analyzed the performance of our model on the test set. We evaluated the performance of our model on samples containing any number of toxic words vs no toxic words. Table 4 shows the results of the analysis. We found that our models performed significantly well for samples having one or more toxic words present and, our best performing model had a perfect F1 score on 66.06 % of them. Our model was unable to find toxic words in only 5.97% of samples containing one or more than one toxic word.

In the case of samples that had no toxic words in a sample, our model could not perform well. Only 6.09% of samples with no toxic words were classified correctly. The dataset statistics for the test set show that samples with no toxic words constitute 19.7 % of the test set. The training and development set had only 6.12% and 6.23% samples without any toxic words. We also found the top 15 most common words which were predicted

as toxic from samples containing no toxic words in the test set. The words are given in Table 5 along with their frequency of occurrence.

We can observe that Table 2 and 5 has common words. We trained our model using token classification objective which tries to capture toxic words. The model cannot identify if the word is part of a toxic/non-toxic sentence. Sometimes these words may be part of a sentence intended to present humor or sarcasm. This may lead the model to incorrectly identify toxic words in samples containing no toxic spans.

## 6 Conclusion

In this paper, we describe our approach for SemEval 2021 Task 5: Toxic Spans Detection. We propose a word-level classifier for identifying the toxic words in a sentence. We experimented with different PLMs to provide a comprehensive analysis of their performance for identifying toxic spans. We performed well, getting a rank of 16 on the leader board. Our analysis shows that a word-level classifier performs extremely well for sentences that contain at least one toxic word. However, it cannot identify cases with no toxic spans efficiently. In the future, we would like to work on solving this problem by using a classifier to simply predict if the sentence is toxic/non-toxic along with span detection.

## References

- Eloi Brassard-Gourdeau and Richard Khoury. 2019. *Subversive toxicity detection using sentiment information*. In *Proceedings of the Third Workshop on Abusive Language Online*, pages 1–10, Florence, Italy. Association for Computational Linguistics.
- François Chollet et al. 2015. Keras. <https://keras.io>.

- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Electra: Pre-training text encoders as discriminators rather than generators](#). In *International Conference on Learning Representations*.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. [Unsupervised cross-lingual representation learning at scale](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.
- Alexis CONNEAU and Guillaume Lample. 2019. [Cross-lingual language model pretraining](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- A. G. D’Sa, I. Illina, and D. Fohr. 2020. [Bert and fasttext embeddings for automatic detection of toxic speech](#). In *2020 International Multi-Conference on: “Organization of Knowledge and Advanced Technologies” (OCTA)*, pages 1–5.
- Paula Fortuna, Juan Soler, and Leo Wanner. 2020. [Toxic, hateful, offensive or abusive? what are we really classifying? an empirical analysis of hate speech datasets](#). In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 6786–6794, Marseille, France. European Language Resources Association.
- Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. [All you need is “love”: Evading hate speech detection](#). In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, AISec ’18, page 2–12, New York, NY, USA. Association for Computing Machinery.
- Taku Kudo and John Richardson. 2018. [SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Keita Kurita, Anna Belova, and Antonios Anastasopoulos. 2019. [Towards robust toxic content classification](#).
- Ping Liu, Wen Li, and Liang Zou. 2019a. [NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Roberta: A robustly optimized bert pretraining approach](#).
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Alex Nikolov and Victor Radivchev. 2019. [Nikolov-radivchev at SemEval-2019 task 6: Offensive tweet classification with BERT and ensembles](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 691–695, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. [Toxicity detection: Does context really matter?](#) In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- M. Schuster and K. Nakajima. 2012. [Japanese and korean voice search](#). In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Neural machine translation of rare words with subword units](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational*

- Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. [SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor!](#) In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 759–773, Barcelona (online). International Committee for Computational Linguistics.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. [Mpnnet: Masked and permuted pre-training for language understanding](#). In *NeurIPS 2020*. ACM.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Shuohuan Wang, Jiaxiang Liu, Xuan Ouyang, and Yu Sun. 2020. [Galileo at SemEval-2020 task 12: Multi-lingual learning for offensive language identification using pre-trained language models](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1448–1455, Barcelona (online). International Committee for Computational Linguistics.
- Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. [UHH-LT at SemEval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1638–1644, Barcelona (online). International Committee for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2020. [Xlnet: Generalized autoregressive pretraining for language understanding](#).
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffenseEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(OffenseEval 2020\)](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

# HLE-UPC at SemEval-2021 Task 5: Multi-Depth DistilBERT for Toxic Spans Detection

Rafel Palliser-Sans and Albert Rial-Farràs

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC - BarcelonaTECH)

Barcelona, Spain

{rafel.palliser, albert.rial}@estudiantat.upc.edu

## Abstract

This paper presents our submission to SemEval-2021 Task 5: Toxic Spans Detection. The purpose of this task is to detect the spans that make a text toxic, which is a complex labour for several reasons. Firstly, because of the intrinsic subjectivity of toxicity, and secondly, due to toxicity not always coming from single words like insults or offends, but sometimes from whole expressions formed by words that may not be toxic individually. Following this idea of focusing on both single words and multi-word expressions, we study the impact of using a multi-depth DistilBERT model, which uses embeddings from different layers to estimate the final per-token toxicity. Our quantitative results show that using information from multiple depths boosts the performance of the model. Finally, we also analyze our best model qualitatively.

## 1 Introduction

SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021) consists in detecting which spans make a text toxic. This is quite relevant for nowadays lifestyle in which, aggravated by the COVID-19 pandemic, online conversations have become key to communicate with our family, friends and job mates, or socialize through social networks and streaming chats. Being able to moderate all this digital content is crucial in order to promote healthy online conversations and discussions.

To tackle this problem, in HLE-UPC we have used a BERT-based model with a fully-connected layer on top to perform Named-Entity Recognition and Classification (NERC), with the goal of tagging each word as either toxic or not. Moreover, we have studied and proved that the use of information from different-depth layers enriches the final classification.

Our contributions to Toxic Spans Detection are:

- The proposal of an ensemble of three different multi-depth DistilBERTs, achieving an F1-score of 68.54% and being ranked 14th out of 91 teams in the challenge, just 2.29% below the best performing model.
- The study of multi-depth BERT-based models in the task of Toxic Spans Detection, showing an improvement on the performance compared to non-multi-depth architectures.
- A qualitative analysis presenting some ethical concerns regarding racial bias.

The source code for our model and pipeline is available at <https://github.com/rafelps/HLE-UPC-SemEval-2021-ToxicSpansDetection>.

## 2 Related work

**Toxicity** The task in which we are participating is not the first one to focus on text toxicity. Without going any farther, in last year's edition of SemEval we can find Task 12, also known as OffensEval 2020 (Zampieri et al., 2020), in which the goal was to identify offensive language in multilingual social media data. In the previous year's competition, SemEval 2019, Task 6 (Zampieri et al., 2019) was also tackling the identification and categorization of offensive language in social media.

Some of the models that solved these tasks involve Convolutional Neural Networks (CNN) (Mahata et al., 2019), Long Short Term Memory Networks (LSTM) (Pham-Hong and Chokshi, 2020) or attention based models (Liu et al., 2019; Wiedemann et al., 2020) branched from the BERT family (Devlin et al., 2019).

**NERC** All the mentioned models approach the task as Sequence Classification, this is, encoding a whole sentence and providing a unique prediction

for it. Toxic Spans Detection, however, goes a step further by asking participants to detect toxic spans, the exact characters or words that make a text toxic. For this reason, instead of modelling the task as sentiment analysis or document/comment classification, it seems more natural to approach it as token classification, generating an output for each token. More specifically, this task could be seen as a Named Entity Recognition and Classification (NERC) task, in which the goal would be to output the most probable sequence of labels (toxic or not) given an input sentence.

In the field of NERC, we also find some interesting models. The state-of-the-art today are attention-based models, usually stemming from transformers such as BERT (Devlin et al., 2019), which can be easily converted into a token classifier by adding a simple linear layer on top of the per-token output. However, we can also find some other attention-based models using CNNs (Baeovski et al., 2019) or even recurrent architectures such as Jiang et al. (2019); Straková et al. (2019); Peters et al. (2018), which mix BiLSTMs, CNNs or CRF layers.

### 3 Data and Methodology

#### 3.1 Data Description

For this task, the organizers provide us with the Toxic Spans Detection (TSD) dataset, also presented in Pavlopoulos et al. (2021), containing phrases and comments that may contain toxic spans. Together with each comment, there is the set of indices of the characters that are considered toxic.

The TSD dataset is split into three subsets: trial, train and test sets with approximately 700, 8000 and 2000 comments respectively. All the models presented in this work have been trained exclusively on the TSD training set, while the trial set has been used to validate our systems. Finally, the test set has served to evaluate the performance of our final models using the available limited submissions for the competition.

The TSD dataset contains very diverse comments. Some of them seem quite simple, but others may be ambiguous, require context knowledge or an understanding of tone, which makes the task extremely challenging. There are also some words that have been written in an ingenious way, to avoid naïve toxic detectors, or that are bleeped or censored. Following we present a couple of examples, where toxic characters are underlined:

- This is a stupid example, so thank you for

nothing a!@#!@.

- I bet you can't wait to see him behind bars.

#### 3.2 Data Cleaning

With a simple data exploration, it can be seen that approximately 90% of the toxic spans exactly match with word boundaries, but in the remaining cases we find strange cases such as the following ones:

1. **You are an idiot:** There is a whitespace as a toxic span boundary.
2. **You are an idiot:** A random singleton character is marked as toxic.
3. **You are an idiot:** “Y” is not marked as toxic but “ou” is.

The majority of these inconsistencies are already known by the organizers of the task and other participants. However, they should still be tackled to provide the best data possible to our models. For this reason, we have cleaned the data using three simple steps and following the idea of toxicity coming from complete words but not from single characters. For each group of consecutive annotated toxic offsets:

1. Iteratively remove the first or last toxic offset if it belongs to a whitespace. This solves the first type of inconsistencies.
2. Remove the toxic offset if it is a singleton: a single consecutive character marked as toxic. This helps in the second type of strange cases.
3. Iteratively left-expand the range of toxic offsets if the previous character is alphanumeric (so it belongs to the same word). Same for right-expansion. This solves the third problem by including the offsets of the whole word as toxic whenever more than one character is marked as so.<sup>1</sup>

After cleaning the data, almost the totality of the annotations matches word boundaries. On one side this confirms our hypothesis that toxicity comes from words or expressions but not from characters. On the other side, this enables a word by word analysis in a consistent and robust manner. Nevertheless, the task remains challenging given the subjectivity of the annotations.

<sup>1</sup>The opposite strategy, discarding words if not all their characters were marked as toxic was also studied but rejected as performed poorer.

### 3.3 Preprocessing

Once data is cleaned and before feeding it to the models, we lower case the text and tokenize it using WordPiece (Wu et al., 2016), the tokenizer used by BERT-based models, which splits text into (usually) sub-word units. Each of these units has its associated token embedding at the first layer of the respective models.

In this step, we also use the information of the already-cleaned toxic offsets to create a per-token binary label regarding its toxicity.

### 3.4 Models

**LSTM** Long Short-Term Memory was introduced in 1991 by Hochreiter and Schmidhuber (1997) as an extension of recurrent neural networks (RNNs), providing them with the ability to capture and memorize long-term dependencies and therefore help prevent the vanishing/exploding problems (Bengio et al., 1994; Pascanu et al., 2013).

We use an LSTM tagger as our baseline model to determine the lower bound performance that we should compare with. We use it as a first approach to solve the task, even though we know that the sentences of the dataset might be too long for the network to memorize and capture all long-term dependencies and the entire sentence context. As input for this model, we use pre-trained word embeddings from GloVe (Pennington et al., 2014).

**Attention-based models** In 2018, Google Research released Bidirectional Encoder Representation from Transformer (BERT) (Devlin et al., 2019) which achieved many state-of-the-art results on different NLP tasks. This success led to the creation of a lot of new models and improvements based on the BERT architecture: DistilBERT, RoBERTa, ALBERT, ... This architecture uses the same multi-head transformer structure presented by Vaswani et al. (2017), which is basically composed of several stacked Transformer blocks/encoders, including self-attention and feed-forward modules. These help the model obtain richer word representations by finding correlations with other tokens in the sentence.

For our task, we use two BERT-based models, BERT and DistilBERT, with a token classification head –a linear layer on top of the hidden state output of the last Transformer encoder–. These models are pre-trained on huge corpus from different sources and fine-tuned for our downstream task.

**Multi-depth models** Based on the previously presented BERT-like models, we implement a modification that consists in feeding the classification layer an augmented embedding for each token. This augmented embedding is formed by concatenating the hidden outputs of different Transformer blocks, instead of using the last output directly as done in common models for token classification. The empirical results show that using embeddings from different layers provides better representations and boosts the model’s performance.

### 3.5 Postprocessing

Once a model outputs its predictions, we loop through them and, for those tokens predicted as toxic, we take their offsets and add them to the final set of toxic spans for that sentence.

Additionally, we add a postprocessing step to increase the correctness of our predictions regarding white characters. These are not returned as tokens by the tokenizer but occupy a character offset. For this reason, for each pair of consecutive tokens predicted as toxic, we also include to the final set the offsets of any white characters in between.

## 4 Results

All the results presented in this section have been calculated using the official metric, the F1-score on the predicted toxic offsets. For detailed information please refer to Pavlopoulos et al. (2021).

### 4.1 Model Comparison

In Table 1 we report the results for the best configuration of each of our models both in the official trial and test sets. In these results, we can first note that all the models clearly outperform our baseline. Moreover, using the information of multiple layers is proved to be beneficial for this task, as it improves each of the respective base models, by 0.64% - 1.42%. Finally, note that although BERT is larger and more powerful than DistilBERT, it performs poorer in the test set. This might be due to the fact that we select our hyperparameters based on the F1-score on the trial set, which is relatively small and may not be representative of the test data. For this reason, our submitted model is a multi-depth DistilBERT, as it provides better generalization within this task and data.

### 4.2 Layer Selection

In this study, we have trained a multi-depth DistilBERT using the outputs of different layers or trans-

Model	F1-score (trial)	F1-score (test)
LSTM (baseline)	61.36%	62.06%
DistilBERT	69.04%	67.43%
BERT	69.22%	66.45%
Multi-depth DistilBERT	69.68%	<b>68.22%</b>
Multi-depth BERT	<b>70.01%</b>	67.87%

Table 1: Performance comparison for various architectures in the official trial (used as validation) and test sets.

former blocks to study its impact on the model’s performance.

Table 2 shows the results for different experiments in which we have concatenated the outputs of the last  $N$  layers of DistilBERT before feeding these enlarged hidden states to the fully connected layer that performs classification.

Results show that performance can be improved by adding different block’s outputs, but can also degrade when using too many. For DistilBERT, which has 6 transformer blocks, the sweet spot seems to be using the last 3 layers. Using all 6 also provides good results, which may imply that the first layer’s output is also quite informative for this task in which words themselves already help predicting their toxicity.

Last N layers	F1-score (trial)
1	69.04%
2	69.48%
3	<b>69.68%</b>
4	69.11%
5	68.94%
6	69.48%

Table 2: Performance comparison for multi-depth DistilBERT in the trial set using the concatenation of the last  $N$  layer’s outputs for the final classification.

### 4.3 Ablation Study

In these experiments, we took apart one component of our system at a time to see its effect on the system’s performance. The main components of our method are presented in Section 3, and details about our implementation can be found in Appendix A.

Table 3 shows the results for this study, in which we can easily see that all components work towards the performance of our model. Apart from the multi-depth component, which has already been studied, Dropout has been key for our giant model to generalize and prevent overfitting the small data.

Using Label Smoothing has also helped, letting the model adapt to the intrinsic subjectivity of the annotations.

Regarding data preparation, it can be seen that the cleaning step has been crucial for the good performance of our system, supporting the known quote “Garbage in, garbage out”. Finally, our simple postprocessing stage has also provided some tenths to the final performance.

Model	F1-score (trial)
Multi-depth DistilBERT	<b>69.68%</b>
(ours) – Multi-depth	69.04%
(ours) – Dropout	68.25%
(ours) – Label Smoothing	69.17%
(ours) – Data Cleaning	66.44%
(ours) – Postprocessing	69.38%

Table 3: Ablation study on the system’s components. ‘–’ means leaving that component out. Results for the official trial set.

### 4.4 Ensemble

Given the results we obtained with single models, we found it interesting to mix some of them to see if they were focusing on different parts of data and could improve the predictions while working together.

Following this idea, we created a simple majority-voting ensemble using the multi-depth models with “last  $N$  layers” for  $N = 1, 3, 6$ ; this is, a base DistilBERT, a model that concatenates the output of the last 3 transformer blocks and another one that uses all 6 layers of DistilBERT.

The final result for this ensemble is 69.34% in the trial set –used as validation– and **68.54%** in the test set, our best submission. Note that although being worse than our best single model in the trial set, it has better generalization skills and boosts the performance in the unseen test set.

## 4.5 Qualitative

Apart from the quantitative analysis done before, we analyze in a qualitative manner the performance and behaviour of our best model, to see how well it detects offensive and toxic words and in which cases it fails.

Below we present some examples of sentences in the dataset together with their ground truth spans and the detection done by the model. The ground truth toxic words appear underlined while the **prediction** is shown in red.

**Correct predictions** We observe how our system is highly capable of identifying toxic and offensive words, both when they appear alone and in multi-word expressions.

- Billy, are you a complete **idiot**, being thick headed or just not reading what people...
- People insist on being **dumb**. No other explanation.
- Could you please **kill yourself**?

**Wrong predictions** However, our system also fails in some challenging comments. As seen below with the word “poorly”, our method misses some words marked as toxic which are not very offensive or disrespectful but can become toxic due to the context.

- People don’t buy that **poorly** built Russian houses...

In other cases, our system identifies toxicity when it is not annotated, although under our perspective the prediction seems correct. This could be due to the ambiguity of the task or inconsistencies in the annotations. An example of it is the expression “freaking donkeys”:

- These **freaking donkeys** all need to be removed from office. I’m so sick and tired of...

Finally, our model fails to detect connectors such as “of” and “and” in between toxic words. In the dataset there are several annotation philosophies: some annotations tend to mark entire expressions as toxic and some others are more word-oriented, excluding connectors between words.

- Are these some of those Russian **pieces of crap** that they seem to be building all over Alaska.

**Ethical concerns** While doing the qualitative analysis we found several examples indicating that there could be racial bias in the predictions of our model, and although it is beyond the scope of the challenge, we found it important to pay attention to it. For this reason, we took some examples from the trial set containing comments about races and changed the words referring to races or origin by others. Below we show an example. The first comment belongs to the competition dataset, while the other is a modification of it, with words “black” changed for “white” and vice versa, and “Mexican” changed for “American”.

- **Black** folks built this nation and got **lynching** for the work. Heck, white folks can be so mean that when they lost their slaves they invited **illegal Mexican immigrants** to do the work black slaves use to do.
- White folks built this nation and got **lynching** for the work. Heck, black folks can be so mean that when they lost their slaves they invited **illegal American** immigrants to do the work black slaves use to do.

We observe that in both cases the system identifies the word “black” as toxic, but not “white”, even when these non-toxic adjectives are the only difference between them. Furthermore, the system only identifies “immigrants” as toxic when appearing next to “Mexican” but not with “American”.

This undesired discrimination happens because there are lots of racist comments in the dataset, which are obviously annotated as toxic. Given that it seems there are more comments against some specific ethnic groups than others, the system associates certain racial references with racism and thus with toxicity.

This is a problem that comes from the data, including the one used in the pre-training phase of BERT models. However, there are several de-bias techniques in the literature (Manzini et al., 2019; Sun et al., 2019; Liang et al., 2020) that could be applied to our model to alleviate it.

## 5 Conclusion

In this work, we have presented a solution for the SemEval-2021 Task 5: Toxic Spans Detection competition, which is a challenging task due to the subjectivity of toxicity and the requirement of context knowledge.



During the development of our solution, a multi-depth DistilBERT model, we have proved the power of pre-trained models and transfer learning to a downstream task with limited data, at the same time that we have demonstrated the benefits of combining the outputs of multiple BERT models' layers for token classification.

With an F1-score of 68.54% the presented model ranks 14 out of 91 participating teams in the competition and, although it presents some racial bias that could be corrected, from the qualitative results we conclude that it has a very good performance, hence being able to be used in real-life applications.

## Acknowledgments

We would like to thank the SemEval-2021 organizers for their effort in preparing the challenge and their help during the competition, as well as the reviewers for their constructive comments. We are also deeply grateful to Jordi Armengol-Estapé from Barcelona Supercomputing Center (BSC) for providing valuable insight into our project.

## References

- Alexei Baevski, Sergey Edunov, Yinhan Liu, Luke Zettlemoyer, and Michael Auli. 2019. [Cloze-driven pretraining of self-attention networks](#).
- Y. Bengio, Patrice Simard, and Paolo Frasconi. 1994. [Learning long-term dependencies with gradient descent is difficult](#). *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 5:157–66.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#).
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural computation*, 9:1735–80.
- Yufan Jiang, Chi Hu, Tong Xiao, Chunliang Zhang, and Jingbo Zhu. 2019. [Improved differentiable architecture search for language modeling and named entity recognition](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3585–3590, Hong Kong, China. Association for Computational Linguistics.
- Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. [Towards debiasing sentence representations](#).
- Ping Liu, Wen Li, and Liang Zou. 2019. [NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 87–91, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Debanjan Mahata, Haimin Zhang, Karan Uppal, Yaman Kumar, Rajiv Ratn Shah, Simra Shahid, Laiba Mehnaz, and Sarthak Anand. 2019. [MIDAS at SemEval-2019 task 6: Identifying offensive posts and targeted offense from Twitter](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 683–690, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Thomas Manzini, Yao Chong Lim, Yulia Tsvetkov, and Alan W Black. 2019. [Black is to criminal as caucasian is to police: Detecting and removing multi-class bias in word embeddings](#).
- Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. [On the difficulty of training recurrent neural networks](#).
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [Semeval-2021 task 5: Toxic spans detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. [GloVe: Global vectors for word representation](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#).
- Bao-Tran Pham-Hong and Setu Chokshi. 2020. [PGSG at SemEval-2020 task 12: BERT-LSTM with tweets' pretrained model and noisy student training method](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2111–2116, Barcelona (online). International Committee for Computational Linguistics.
- Jana Straková, Milan Straka, and Jan Hajic. 2019. [Neural architectures for nested NER through linearization](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331, Florence, Italy. Association for Computational Linguistics.
- Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. [Mitigating gender bias in natural language processing: Literature review](#).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. [UHH-LT at SemEval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1638–1644, Barcelona (online). International Committee for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google’s neural machine translation system: Bridging the gap between human and machine translation](#).

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(offenseval\)](#).

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(offenseval 2020\)](#).

## A Implementation Details

We have developed this project using PyTorch<sup>2</sup> and the Huggingface<sup>3</sup> implementation of transformers. Specifically, we have used BertModel and DistilBertModel pre-trained models.

Although Huggingface includes models for token classification, the input dimension for their classification layer is exactly 768 (for BERT and DistilBERT), the dimension of the Transformer blocks’ output. However, in our case, we are concatenating different outputs, so these dimensions will vary from one experiment to another. To mimic their token classification models, we manually add a Dropout layer and the final classification layer with the appropriate input dimension (in our case  $768 \times \text{\#concat\_outputs}$ ).

To train the models we use Cross Entropy Loss with Label Smoothing. This type of regularization slightly changes the target vector, asking the model to predict  $1 - \epsilon$  for the correct class and  $\epsilon$  for the others instead of the usual hard assignment of 1 for the true class. As seen in Section 4.3, this technique helps the system improve as it helps modelling the intrinsic subjectivity of the data. For our best models we use  $\epsilon = 0.1$ .

We perform from 4 to 8 training epochs with Adam optimizer, learning rate  $10^{-5}$ , batch size of 8 and 25% dropout rate. Finally, we select the epoch with the best F1-score at trial set as our best checkpoint.

We keep the default values for the rest of parameters.

Each model has required approximately 15 minutes of training time on an NVIDIA Tesla V100 GPU. With the same hardware, the inference time is 430ms per sentence, so our system is able to work in Near Real Time (NRT).

---

<sup>2</sup><https://pytorch.org/>

<sup>3</sup><https://huggingface.co/transformers/>

# Lone Pine at SemEval-2021 Task 5: Fine-Grained Detection of Hate Speech Using BERToxic

Yakoob Khan, Weicheng Ma, Soroush Vosoughi

Department of Computer Science, Dartmouth College

{yakoob.khan.21, weicheng.ma.gr, soroush.vosoughi}@dartmouth.edu

## Abstract

This paper describes our approach to the *Toxic Spans Detection* problem (SemEval-2021 Task 5). We propose **BERToxic**, a system that fine-tunes a pre-trained BERT model to locate toxic text spans in a given text and utilizes additional post-processing steps to refine the boundaries. The post-processing steps involve (1) labeling character offsets between consecutive toxic tokens as toxic and (2) assigning a toxic label to words that have at least one token labeled as toxic. Through experiments, we show that these two post-processing steps improve the performance of our model by 4.16% on the test set. We also studied the effects of data augmentation and ensemble modeling strategies on our system. Our system significantly outperformed the provided baseline and achieved an F1-score of 0.683, placing Lone Pine in the 17<sup>th</sup> place out of 91 teams in the competition. Our code is made available at <https://github.com/Yakoob-Khan/Toxic-Spans-Detection>

## 1 Introduction

The promotion of respectful discourse has always been a core tenet of civilized societies. The Cambridge dictionary defines hate speech as “public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation.” Online platforms enable malicious actors to hide behind a cloak of anonymity and surreptitiously post toxic comments that are a “menace to democratic values, social stability and peace” (United Nations). To combat this problem, such platforms often employ human moderators to address offensive content that goes against community standards. However, moderators are unable to manually keep pace with the large volume of user-generated content today. This motivates the development of natural language processing systems to automatically detect hate speech

and ensure that online platforms remain healthy and inclusive for all.

There has been extensive research on hate speech detection, with the creation of large datasets (Wulczyn et al., 2017) and the use of pre-trained text representations (Devlin et al., 2019) for varied modeling approaches. Competitions on offensive language identification (Zampieri et al., 2020) have further attracted attention to this topic. Prior work has hitherto focused on classification at the document-level based on various taxonomies, such as whether a given text contains offensive language or if it is targeted towards an individual or group. This line of inquiry does not identify the toxic spans that ascribe a text as hate speech. Doing so will assist human moderators to efficiently locate offensive content in long posts and elucidate further insight into hate speech explainability.

This motivated the *Toxic Spans Detection* task (Pavlopoulos et al., 2021) where systems are asked to extract the list of toxic spans that attribute to a text’s toxicity. Consider the following example<sup>1</sup>:

<b>Text</b>	Because he’s a <b>moron</b> and <b>bigot</b> . It’s not any more complicated than that.
<b>Span</b>	[15, 16, 17, 18, 19, 27, 28, 29, 30, 31]

Table 1: A sample example from the task dataset.

As there are two toxic spans in the above text, systems are tasked to extract the character offsets (zero-indexed) corresponding to the sequence of toxic words. This is a challenging task as classification at the word-level is inherently more difficult than at the document-level. The intentional obfuscation of toxic words, use of sarcasm and the

<sup>1</sup>We caution readers that the examples included in this work contain explicit language to illustrate the severity and challenges of hate speech detection.

subjective nature of hate speech further adds complexity to the problem.

Our contributions to the task is threefold:

1. We propose **BERToxic**, a system that fine-tunes a pre-trained BERT model with additional post-processing steps to achieve an F1-score of 0.683, placing Lone Pine in the 17<sup>th</sup> place out of 91 teams in the competition.
2. We study the effects of simple data augmentation strategies on our system and find that they yield no improvement in classification performance.
3. We examine late fusion and multi-task learning neural architectures and conclude that they under-perform compared to the standalone BERT model for this task.

## 2 Our Approach

### 2.1 Baselines

To have a better sense of our final system’s performance, we initially examined two baseline models. First, we created a trivial model that randomly predicts each character offset of a text as toxic if its  $\rho > 0.5$ , drawn from a continuous uniform probability distribution.

To have a stronger baseline model, we fine-tuned the off-the-shelf spaCy NER model provided by the task organizers. This model consists of a multi-hash embedding layer (feed-forward sub-network) that uses sub-word features and an encoding layer consisting of a CNN and a layer-normalized max-out activation function. The model uses a transition-based algorithm that assumes that the “most decisive information” regarding the entities “will be close to their initial tokens”, with a loss function that optimizes for whole-entity accuracy.

### 2.2 BERToxic

We framed the toxic spans detection task as a sequence labeling problem and leverage the **B**idirectional **E**ncoder **R**epresentation from **T**ransformers (Devlin et al., 2019) model to extract rich feature representations from the input texts.

The first step in the BERToxic system pipeline (Figure 1) was to tokenize the text inputs and generate the word embeddings using BERT’s WordPiece tokenizer. This sub-word tokenization algorithm (Schuster and Nakajima, 2012) tokenizes a word like "moron" into ["mo", "##ron"] and we

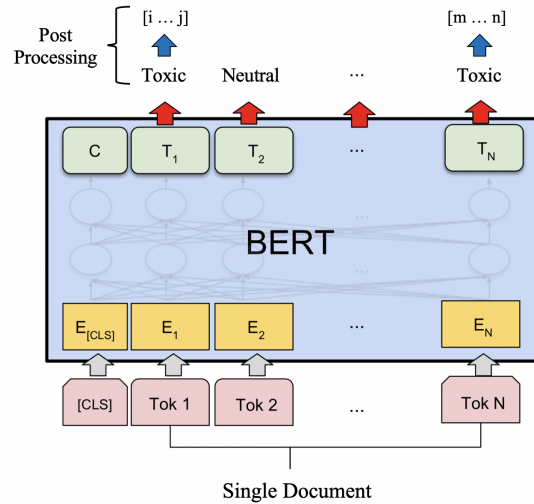


Figure 1: The BERToxic model architecture. Image modified from (Devlin et al., 2019).

ensured that the ground truth labels were preserved across all tokens of a word. As BERT uses absolute position embeddings, we padded shorter sequences with [PAD] tokens on the right side such that all tensor inputs are set to equal the maximum sequence length observed for batched parallelized training. Long sequences were truncated to 512 tokens, the maximum sequence length allowed by BERT. As the data was obtained from online comments that are generally shorter in nature, the truncation procedure was not needed in this task but nevertheless served to handle long sequences if present. We also stored the mapping

$$\mathcal{M} : t_i \mapsto (start_i, end_i)$$

of each token to its relative character offsets in the original string, used for outputting the toxic span predictions at the post-processing stage.

We performed all of our experiments using the BERT<sub>BASE</sub> model architecture that consisted of 12 layers, 768 hidden size, 12 self-attention heads and 109M parameters. The BERT<sub>LARGE</sub> model was not explored in this work due to its compute-intensive nature. Our intuition suggested that letter casing could be helpful for this task as proper nouns (e.g *Muslim*) can be used offensively, so we selected the cased model for our experiments. A token classification head containing a linear layer was applied on top of the final hidden-states output, with a label prediction of 1 denoting a toxic token, 0 otherwise. For each token  $t_i$  labeled as toxic, we utilized  $\mathcal{M}$  to output all character indices in the range of  $(start_i, end_i)$  inclusive as the toxic span

of this token.

Additionally, our system performed two post-processing steps to refine the boundary predictions. Consider the following tokenized sequence:

$$t_1, \dots, t_i, t_{i+1}, t_{i+2}, \dots, t_n$$

First, for any two consecutive tokens  $t_i$  and  $t_{i+1}$  whose prediction labels are toxic, we output the character indices in the range of  $(end_i + 1, start_{i+1} - 1)$  inclusive as toxic as well. This had the effect of including the delimiter characters between consecutive toxic words, thereby detecting toxic phrases. Second, recall that BERT’s WordPiece tokenizer could split a word into multiple tokens, say  $t_i, t_{i+1}$  and  $t_{i+2}$ . If at least one token was predicted toxic by the model, our system assigned a toxic label to all constituent tokens of this word. This achieved coherence in the prediction of toxic words and phrases, thus avoiding incomplete word piece issues.

We also attempted to vary the thresholds of the confidence scores before SoftMax for toxic token predictions but observed no improvement in performance.

### 2.3 Data Augmentation

Data augmentation is widely used to improve the generalization of models by acting as a regularizer to reduce overfitting. While various sophisticated techniques exist to artificially enhance the size and quality of the training set without collecting additional manually labeled examples, we chose to apply the set of **Easy Data Augmentation** (EDA) techniques (Wei and Zou, 2019) to generate synthetic training data for this task.

The four operations in EDA are Synonym Replacement (SR) using WordNet (Miller, 1995), Random Insertion (RI), Random Swap (RS) and Random Deletion (RD) of words in a document. Shorter documents are disproportionately more affected by these operations if a fixed number of words are modified per document. To ensure that all documents experienced the augmentation strength proportionately, the number of words  $n$  modified was varied based on the document length  $l$  using the formula  $n = \alpha \cdot l$ , where  $\alpha$  is a hyperparameter that indicates the percentage of words changed per document. Each operation was applied once per document and care was taken to ensure that the ground truth labels were preserved.

Our experiments revealed that the recommended value of  $\alpha = 0.1$  was too low for this task and

we observed small but consistent improvements as  $\alpha$  increases. Furthermore, we noted that the SR technique alone leads to better performance than using all four operations to create the augmented training set for this task.

We also attempted data augmentation using an external dataset, HateXplain (Mathew et al., 2020), that contains 20,148 documents with word-level annotations that we processed to conform to this task’s data format. Each document consisted of 2-3 annotations and we used their intersection to maximize the inter-annotator agreement in constructing the ground truth labels. HateXplain’s annotation strategy appeared to be different and included labeling pronouns, conjunctions and stop words as toxic when located between offensive words. We removed such toxic labels so that the external dataset annotation was more similar to this task. When our task dataset was augmented with the full external dataset, the model experienced underfitting, while removing all the non-toxic labeled documents from the external dataset alleviated the issue to some extent.

### 2.4 Ensemble Modeling

Ensemble modeling is an approach where multiple different models are trained and their predictions are aggregated. By adding bias to counter the variance of a single model, this line of work has been shown to improve the predictive performance of a system (Liu et al., 2019). While numerous ensemble modeling techniques like boosting, bagging, etc. exist, we investigated two techniques of interest: late fusion and multi-task learning.

We reframed the problem as a binary classification task and trained a sequence classifier to predict whether a given sentence is toxic. In the late fusion approach, we utilized NLTK’s tokenizer to split each document into sentences. If a sentence contained a ground truth toxic span, we assigned the toxic class label 1, 0 otherwise. In this way, a binary classification dataset was created to separately fine-tune a pre-trained BERT sequence classifier. We hypothesized that token labels should be predicted toxic only if the corresponding sentence was classified as toxic as well. Late fusion was performed at the prediction phase, where both the sequence and token classifiers voted in the predictions by having the former model filter toxic sentences on which the latter model made final toxic span predictions.

Rather than fine-tuning the two models separately, we also investigated if multi-task learning (MTL) improved the predictive performance of the ensemble model. We hypothesized that a training regime where the two classifiers were learned jointly could be useful as the knowledge gained in learning one task could benefit the other. To perform MTL, we fine-tuned an MT-DNN (Liu et al., 2019) model where the text encoding lower BERT layers are shared across the two tasks while the top layers are task-specific.

### 3 Experiments

The following sections describe the experimental set-up of our work.

#### 3.1 Dataset

The task data was sourced from the Civil Comments dataset (Borkan et al., 2019), which contains public comments made between 2015 - 2017 that appeared on approximately 50 English-language news sites across the world. As the original dataset contained only document-level class labels, the task organizers selected a subset of the data for crowd-sourced toxic spans annotation. For the data split, we chose to fine-tune our models using the entire provided training dataset ( $N = 7939$ ) to maximize performance, validate using the trial dataset ( $N = 690$ ), and submit our predictions using the test data ( $N = 2000$ ). The test labels were withheld during the evaluation phase of the competition and were only released afterward.

#### 3.2 Evaluation Metric

To evaluate the performance of the models, the task organizers employed a variant of the F1-score (Da San Martino et al., 2019). For a document  $d$ , define  $S_d$  as the set of toxic character offsets predicted by a system and  $G_d$  as the set of ground truth annotations. Then the F1-score of the system with respect to ground truth  $G$  for  $d$  is defined as

$$F_1^d(G) = \frac{2 \cdot P^d(G) \cdot R^d(G)}{P^d(G) + R^d(G)}$$

where

$$P^d(G) = \frac{|S_d \cap G_d|}{|S_d|}$$

$$R^d(G) = \frac{|S_d \cap G_d|}{|G_d|}$$

If a document has no ground truth annotation ( $G_d = \emptyset$ ), or the system outputs no character offset

prediction ( $S_d = \emptyset$ ), we set

$$F_1^d(G) = \begin{cases} 1 & G_d = S_d = \emptyset \\ 0 & otherwise \end{cases}$$

We finally take the arithmetic mean of  $F_1^d(G)$  over all the documents of an evaluation dataset to obtain a single F1-score for the system.

### 3.3 Implementation Details

We utilized the PyTorch framework for the development of our system, HuggingFace’s transformers library (Wolf et al., 2020) for the BERT-based models and Microsoft’s implementation of the MT-DNN model. All models were trained on Google Colab Pro’s High-RAM environment using a single NVIDIA P100 GPU. The training policy used the following hyper-parameters: batch size of 16, sequence length of 512, weight decay of 0.01. For optimization, we used Adam with a learning rate of  $5e-5$  and a linear warm-up schedule over 500 steps. Except for MT-DNN<sup>2</sup>, all our models were fine-tuned for approximately 2 epochs and we practiced early stopping by monitoring the dev F1-score to reduce overfitting. The EDA experiment was performed with  $\alpha = 0.8$  using only the SR technique. All other hyper-parameters were set to their default values according to HuggingFace’s implementation. We set a random seed for all our experiments and open-sourced the code for reproducibility.

## 4 Results

On the following page, Figure 2 visualizes the precision-recall curves<sup>3</sup> of all the models and Table 2 summarizes their performance metrics. Figure 3 shows the confusion matrix of our best performing BERToxic model and Table 3 highlights selected predictions that it made.

An interesting observation we noted from Table 2 was that the F1 scores for the test set were higher than the dev set for many of the models. We hypothesize that this is because the models have an inductive bias to predict shorter toxic spans, evidenced by the average ground truth span length of 7.2 in the test set and 14.7 in the dev set.

<sup>2</sup>MT-DNN was fine-tuned for 3 epochs with a batch size of 8.

<sup>3</sup>The curves for the spaCy and BERT multi-task model are less detailed due to the ambiguity in obtaining the probability scores from their respective implementations, necessitating the use of their predicted labels instead.

Model	Dev			Test		
	Precision	Recall	F1	Precision	Recall	F1
Random	0.143	0.463	0.175	0.089	0.413	0.122
SpaCy	0.692	0.588	0.595	0.664	0.686	0.656
BERToxic	0.781	0.678	0.681	0.683	0.732	<b>0.683</b>
+ EDA	0.787	0.683	0.684	0.681	0.725	0.678
+ HateXplain	0.792	0.674	0.681	0.683	0.721	0.678
BERT late fusion	0.733	0.636	0.639	0.675	0.709	0.669
BERT multi-task	0.744	0.629	0.634	0.665	0.694	0.656

Table 2: A summary of the performance of all our models, reporting the precision and recall scores along with the F1 evaluation metric used for the competition. The BERToxic model outperformed the strong spaCy baseline by 4.16% on the test set, placing Lone Pine in the 17<sup>th</sup> place out of 91 teams. In comparison, the top-ranked submission achieved an F1-score of 0.708. The experiments revealed that our data augmentation and ensemble modeling strategies did not outperform the standalone BERT model.

Our proposed system performed well at the toxic spans detection task, showing strength in identifying profanity and common toxic words like “idiot” and “stupid”. The model identified the obfuscation of offensive words and successfully detected hate speech from such adversarial cases (Example 1).

1. Kill this *F'n W\*ore* on site.
2. .. how I am an *ignorant fool* ..
3. *Nazi boneheads* deserve being *punched*.
4. @remoore *Shut up, racist*.
5. *Cruz is a piece of garbage a globalist fraud*

Table 3: Selected examples obtained from the test set. BERToxic’s predictions are shown in red while ground truth annotations are *italicized*.

The error analysis revealed that the system lacked nuance as it would sometimes classify toxic words used in neutral contexts (Example 2). It is also worth mentioning that there was considerable noise in the ground truth annotations. Our manual inspections concurred with the model’s predictions that some words and phrases were used in offensive contexts but the annotators thought they were neutral (Example 3 and 4). Furthermore, we observed some inconsistencies in the labeling scheme as some annotations spanned entire sentences (Example 5) while others only highlighted a few words in the sentence. These issues point to the subjective nature of hate speech and the challenges involved in its fine-grained classification.

We found through our ablation studies of data augmentation that generating synthetic data using

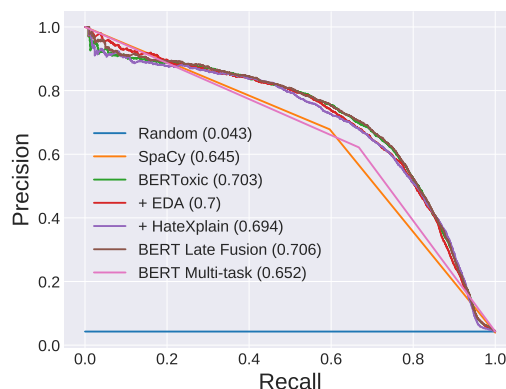


Figure 2: Comparison of the precision-recall curves of all the models at the token level on the test set. The area under the curve is enclosed within parentheses.

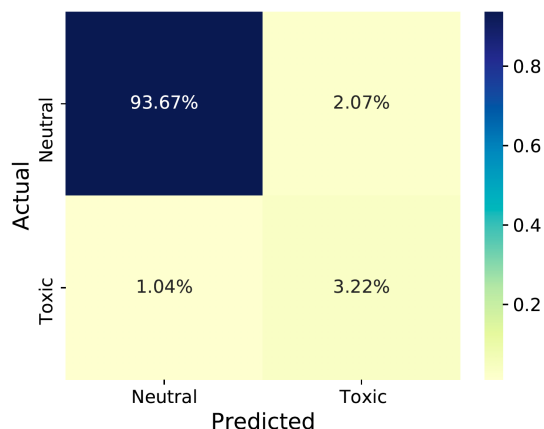


Figure 3: A confusion matrix of the BERToxic system at the token level, revealing insights about the classification performance in each category and highlighting the imbalance of the class labels in the test set.

the EDA techniques did not improve the performance of the system. This suggested that the dataset size does not appear to be the limiting factor affecting the performance of BERT in this task. Using HateXplain’s external dataset, we learned that different data sources and annotation guidelines can introduce noise that hurts the performance of models.

Finally, the ensemble modeling strategies we explored did not outperform the standalone BERT model. The late fusion technique performed slightly better than the spaCy baseline, but it seemed that the sequence classifier made errors on similar parts of the input space as the token classifier. The multi-task learning approach underperformed compared to late fusion, suggesting that the sequence labeling and classification tasks are not closely related enough to benefit their joint training.

## 5 Conclusion and Future Work

In this work, we have proposed BERToxic, an empirically powerful system that performed fine-grained detection of hate speech. We found that our exploration of data augmentation and ensemble modeling strategies did not outperform the standalone model. The error analysis revealed that BERT lacked nuance in understanding the use of offensive words in neutral contexts and encountered boundary detection issues when faced with noisy ground truth annotations.

Future avenues of work could address these limitations and explore other transformer-based models to develop more robust hate speech detectors. We hope that our findings inspire more creative approaches towards fine-grained detection of hate speech so that online discourse can remain healthy and inclusive for all.

## Acknowledgments

Yakoob Khan is thankful to be supported by the Stamps Scholarship funded by Dartmouth College and the Strive Foundation.

## References

Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav

Nakov. 2019. [Fine-grained analysis of propaganda in news article](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. [Multi-task deep neural networks for natural language understanding](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496, Florence, Italy. Association for Computational Linguistics.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. [Hatexplain: A benchmark dataset for explainable hate speech detection](#). *arXiv preprint arXiv:2012.10289*.

George A. Miller. 1995. [Wordnet: A lexical database for english](#). *Commun. ACM*, 38(11):39–41.

John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. [SemEval-2021 task 5: Toxic Spans Detection \(to appear\)](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

M. Schuster and K. Nakajima. 2012. [Japanese and korean voice search](#). In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

Jason Wei and Kai Zou. 2019. [EDA: Easy data augmentation techniques for boosting performance on text classification tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.



Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#). In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1391–1399, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(OffenseEval 2020\)](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

# SRPOL DIALOGUE SYSTEMS at SemEval-2021 Task 5: Automatic Generation of Training Data for Toxic Spans Detection

**Michał Satława, Katarzyna Zamłyńska, Jarosław Piersa,  
Joanna Kolis, Klaudia Firląg, Katarzyna Beksa,  
Zuzanna Bordzicka, Christian Goltz, Paweł Bujnowski**

Samsung R&D Institute Poland

{m.satlawaw;k.zamlynska;j.piersa}@samsung.com  
joanna.kolis@gmail.com, {k.firlag;k.beksa}@samsung.com  
{z.bordzicka;c.goltz;p.bujnowski}@samsung.com

**Piotr Andruszkiewicz**

Samsung R&D Institute Poland  
Warsaw University of Technology  
p.andruszki2@samsung.com

## Abstract

This paper presents a system used for SemEval-2021 Task 5: Toxic Spans Detection. Our system is an ensemble of BERT-based models for binary word classification, trained on a dataset extended by toxic comments modified and generated by two language models. For the toxic word classification, the prediction threshold value was optimized separately for every comment, in order to maximize the expected F1 value.

## 1 Introduction

Freedom of speech is one of the most important human rights. However, because the definition of protected speech is not precise enough, it can be easily misinterpreted and misused. The problem is magnified in cyberspace, where anonymity and asynchronous communication contribute to toxic disinhibition. As a result, the Internet has become space where hatred, harsh criticism, rude language and threats may grow.

Currently, identification of such harmful content may depend mostly upon classification models that detect abusive comments or documents. However, SemEval-2021 Task 5: Toxic Spans Detection proposes detecting fragments of text that make it toxic, with the aim of supporting manual moderation of oftentimes lengthy comments. A successful solution to this problem would be a crucial step towards more constructive and inclusive online discussions. The task focuses on English, which is the most common language used on the Internet, as of January 2020 (Johnson, 2021).

In this paper we present the model we used for toxic span detection, the method we used to find optimal prediction threshold values, and two methods for producing new training examples with toxic spans annotation:

- Resampling the data: new examples are generated by substituting non-toxic words with predictions from a language model.
- Data generation: we trained a simple language model on existing examples, in order to generate new examples containing marked toxic spans.

A total of 91 teams made an official submission on the test set with the best submission achieving F1 score of 0.7083. Our approach was ranked as 11th with 0.6865 F1 score (for details see Section A of the Appendix).

## 2 Related Work

The interest in automatic identification of abusive language has increased among researchers due to the importance of public discussion in the Internet and its public impact. To our mind, the domain overlaps with other NLP tasks, such as sentiment analysis or comment classification.

In an earlier piece of work, (Yin et al., 2009) studied harassment detection on Web 2.0 datasets (i.e. Kongregate, Slashdot and MySpace) using TFIDF n-gram features and SVM models. In other research, (Sood et al., 2012a,b) analysed profanity detection in a community-based news site Yahoo!

Buzz with SVM and Levenshtein distance. In later studies, (Wulczyn et al., 2017) tried to understand personal attacks in English Wikipedia discussion comments using logistic regression and multilayer perceptron classifiers on character and word n-grams.

In the last years there were several contests focusing on various aspects of offensive language. One of them included hate speech (Bosco et al., 2018) and misogyny identification (Fersini et al., 2018) for Italian data. Another workshop concerning a similar topic was Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter at SemEval-2019 (Garibo i Orts, 2019). Recently, research has focused mostly on deep learning classification of cyber hate using "accept"/"non-offensive" or "reject"/"offensive" classes, e.g. (Pavlopoulos et al., 2017). However, more and more papers concern explainability components and reasons. For example, organizers of SemEval-2019 Task 6, (Zampieri et al., 2019) required identification of the offensive content type and the target of the offensive post. In another study, carried out by (Mathew et al., 2020), aside from simple performance metrics, more complex bias and explainability factors were used to evaluate various deep neural networks models (CNN-GRU, BiRNNs and BERT). A similar idea – to clarify rationales for hate speech – comes in SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021). Our solutions to this challenge are presented in this paper.

### 3 Data

There are several abusive language detection datasets available (such as (Wulczyn et al., 2017; Borkan et al., 2019)). However, their purpose is toxicity detection of the whole text, so they do not contain information about the exact spans that make a text toxic. For such a task, the SemEval-2021 Toxic Spans Detection dataset was created. It consists of 10,629 English texts and their relative lists of toxic spans' indices (7,939 train, 690 trial, and 2,000 test texts with their respective spans). Examples from the dataset are presented in Table 1.

The SemEval-2021 Task 5 annotated data turned out to be a challenging material to work on in some respects.

It results mainly from somewhat inconsistent annotation which is visible in several dimensions, see Appendix B.

## 4 System Overview

### 4.1 Resampled Data

The texts from train dataset were tokenized and the tokens that were not labeled as toxic spans were replaced with tokens suggested by a RoBERTa (Liu et al., 2019) language model. We set the limit of substituted words on 1/3 of all tokens outside the toxic span, but no more than 10. The replacement was not applied to the punctuation. Also, it was not possible to change the word to the same one, or to the word preceding the token in question. The tokens to be substituted are chosen randomly so it was possible to obtain different outcomes while processing the same text several times:

original text:

*Uh, yes he is **stupid**...as his actions have proven on countless occasions...*

generated:

*Uh, Donald Trump 's **stupid**...as recent history has shown in numerous occasions...*

*Uh, now that 's **stupid**...as my students proudly demonstrated in numerous occasions...*

*Uh, well that 's **stupid**...as Vikings fans themselves said in numerous occasions...*

### 4.2 Generated Data

The second method of expanding the original training dataset was inspired by the Ding et al. (2020) data augmentation method for the POS tagging. Firstly, the pairs of spans indices and texts from training data were encoded in linearized form, e.g. the original span and text pair:

"[9, 10, 11, 12, 13]", "One less **idiot** to worry about."

was transformed into text

"<s> One less <toxic> idiot </toxic> to worry about. </s>"

where <toxic>, </toxic> are special tokens which indicate the beginning and ending of a toxic span and <s>, </s> are special tokens indicating that this particular example is annotated. The set of such linearized training examples was extended by a dataset of the same size, containing unannotated examples from Civil Comment Dataset (Borkan et al., 2019) (with a different special starting token <u>). These examples were used to train a simple language model with 300-dimensional word embedding and two 256-dimensional LSTM layers. This language model was used to generate new sentences in linearized format, starting from the token <s> and randomly sampling next words from the

Text	Toxic span
What a pile of shit. I love Bruce and I could write a better case against Bruce than this rubbish !	[15, 16, 17, 18, 90, 91, 92, 93, 94, 95, 96]
What a jerk!	[7, 8, 9, 10]
Deep ecology madness is a sad sick religion.	[]

Table 1: SemEval-2021 Task 5 dataset examples.

model distribution until the ending token  $\langle /s \rangle$  is sampled or the generated text length exceeds 200 tokens. 10 000 of such generated examples were converted back into the format span + text and used for training the token classifier model.

### 4.3 Token Classification

To detect toxic spans within a text we use the Hugging Face (Wolf et al., 2019) implementation of the BERT model (Devlin et al., 2018). We approach the task as token classification. Therefore, there are additional layers on top of the model – a classification head (a one-dimensional linear layer on top of the hidden-states output), preceded by a 50% dropout layer.

### 4.4 Character Classification

The metric used for competition ranking was the mean value of character-level F1 score, as in (Da San Martino et al., 2019), so the token-level predicted probabilities from our model needed to be converted to character-level binary labels. The process included two stages.

- Assigning probabilities to text characters. Every character that is a part of a token is assigned the predicted probability of that token. All the other characters (e.g. whitespaces) are assigned 0.
- Choosing and applying the optimal threshold value for a given text example, based on the predicted characters probabilities. Characters with probabilities meeting this threshold are identified as being part of a toxic span.

The threshold value was optimized separately for every text example to maximize the expected value of character-level F1 score, where the F1 value for a predicted span is a random variable with respect to the distribution of golden spans.

Formally, for a given sentence composed of  $n$  letters, let's denote predicted labels (0 and 1) as  $s = (s_1, \dots, s_n)$  and golden labels as  $y = (y_1, \dots, y_n)$ .

Then the F1 measure for such prediction is

$$F_1(s, y) = \frac{2 \sum_{i=1}^n s_i y_i}{\sum_{i=1}^n s_i + \sum_{i=1}^n y_i}$$

If elements of  $s$  and  $y$  are indexed in order of decreasing probabilities of being toxic, and  $S_{i:j} = \sum_{k=i}^j y_k$ , then the expected value of  $F_1$  for the prediction with  $k$  most probable letters classified as toxic, is

$$f(k) = \sum_{k_1=0}^k \sum_{k_2=0}^{n-k} \frac{2P(S_{1:k}=k_1)P(S_{k+1:n}=k_2)k_1}{k + k_1 + k_2}$$

We approximated the distribution of the golden spans  $y$  by assuming that probabilities of characters in the golden toxic span are independent and equal to probabilities predicted by the model. This allowed us to use  $O(n^2)$  algorithm proposed by (Nan et al., 2012) to find the  $k$  which maximizes the  $f(k)$ .

### 4.5 Ensemble Models

On top of the plain prediction models we applied a selection of ensembles (Opitz and Maclin, 1999). Ensembles are aggregations of models solving the same problem, built with the hope that a panel of experts can give a better decision than a single one. In our case, the ensemble accepted character-level inputs from 2 to 9 participating models and returned a single character-level output. Among the available range, we used:

- set-theory union (i.e. at least one model declared a character as toxic),
- set-theory intersection (i.e. all models declared a character as toxic),
- majority voting (i.e. at least half of the models declared a character as toxic),
- F1-weighted voting (i.e. the vote was weighted with the model's F1 score and computed for the evaluation set),

The output of the ensemble still needed to be post-processed.

## 5 Experimental Setup

The officially released datasets (both train and trial) contained altogether 8,629 texts. From those datasets we generated nine random train/dev/test splits, with the ratio 80/10/10%.

From the train sets in each of the splits new data was generated using both methods described in 4.1: resampling data outside the span, and data augmentation. For resampling data, we used the BERT based model (uncased) model with BertTokenizerFast tokenizer or RoBERTa base model with RobertaFastTokenizer. The upper limit of changes in the text was equal to 10.

We trained the token classification model using only the train set, as well as the train set together with one or two sets of generated data. We used binary cross-entropy loss function and early stopping technique, ending the training after 3 consecutive epochs without the decrease of loss function on the dev set.

The other hyperparameters used in the training were: batch size: 8, dropout rate: 0.5, learning rate:  $1e-05$ , and max token sequence length: 340. We wanted to compare models trained on dataset with diverse sizes and, due to resampling, a lot of similar examples. To keep the evaluations for the early stopping in short and uniform intervals across all models we set a fixed number of steps per epoch: 800.

All the nine models obtained from given cross-validation splits were used for ensemble models. The best results were obtained via intersection and majority voting over models trained on cross-validation splits.

The final stage was postprocessing the spans. Whitespaces and punctuation characters that were the only characters separating two parts of toxic spans were included in the toxic span, while all the other whitespaces and punctuation characters located at the ends of spans were removed.

## 6 Results

The results of the models submitted to SemEval-2021 competition are presented in Table 2. All of the three models were trained on the official dataset and on data generated with resampling method. The results shows that adding more components of our system improves the final result. The first model is only token classifier model trained on enlarged dataset. When it comes to the second one, we used models trained on 9 different train/dev splits and

aggregated their results using ensemble. The last one was improved by the threshold optimization and additional data generated with language model.

We checked the results of token classifier trained with data obtain with the other tokenizers available, see Table 3. Our best result was obtained for XLMRobertaTokenizerFast and it exceeded our best result in the competition.

The results of the models trained on augmented datasets, either with or without optimization of the prediction threshold, are presented in Table 4. It can be observed that adding more noisy and automatically generated data to the training worsened the model results, making them more variable. The addition of the threshold that optimized the expected value of F1 metric independently on every test set example fixed both issues, producing models with higher and more stable results.

## 7 Conclusions

In many classification tasks we can observe a divergence of the objective function (e.g. F1 score), optimized loss function (e.g. cross-entropy) and applied prediction thresholds. Our results demonstrate that even when the distribution of golden classes is crudely approximated by the assumption of independent and underperforming underlying model, the F1-optimized threshold values perform better than commonly used and accuracy-optimized threshold of 0.5 in the setting with noisy and automatically augmented training data.

The implemented method of data augmentations, based on resampling non-toxic words proved to be effective by increasing the F1 score of token classifier.

Model	data augmentation	F1 on test data
BERT	resampled	0.6826
intersection ensemble over cv splits	resampled	0.6847
voting ensemble over cv splits, threshold optimization	resampled, generated	<b>0.6865</b>

Table 2: Results of our top models submitted to competition.

language model	tokenizer	F1 on trial set	F1 on test set
xlm-roberta-base	XLMRobertaTokenizerFast	0.6732	<b>0.6910</b>
facebook/bart-base	BartTokenizerFast	0.6610	0.6832
bert-base-uncased	BertTokenizerFast	0.6999	0.6684
bert-large-uncased	BertTokenizerFast	0.7007	0.6700
google/electra-large-generator	ElectraTokenizerFast	0.6984	0.6735

Table 3: Results for different tokenizers used in resampled data generation. The names of models and tokenizers are taken from <https://huggingface.co/models>.

data augmentation	prediction threshold	F1 mean	F1 std. deviation
resampled, generated	F1-optimized	<b>0.6700</b>	0.0115
	0.5	0.6643	0.0128
resampled	F1-optimized	0.6670	<b>0.0093</b>
	0.5	0.6644	0.0148
generated	F1-optimized	0.6643	0.0110
	0.5	0.6666	0.0105
none	F1-optimized	0.6664	0.0121
	0.5	0.6688	0.0107

Table 4: The results obtained using cross validation split and voting ensemble for different datasets. The token classifier was trained with F1-optimized threshold as well as fixed threshold.

## References

- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019. *Nuanced metrics for measuring unintended bias with real data for text classification*. *CoRR*, abs/1903.04561.
- Cristina Bosco, Felice Dell’Orletta, Fabio Poletto, Manuela Sanguinetti, and Maurizio Tesconi. 2018. *Overview of the EVALITA 2018 Hate Speech Detection Task*, pages 67–74. Torino: Accademia University Press.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. *Fine-grained analysis of propaganda in news article*. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. *BERT: pre-training of deep bidirectional transformers for language understanding*. *CoRR*, abs/1810.04805.
- Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. *DAGA: Data augmentation with a generation approach for low-resource tagging tasks*. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057, Online. Association for Computational Linguistics.
- E. Fersini, P. Rosso, and M. Anzovino. 2018. *Overview of the task on automatic misogyny identification at ibereval 2018*. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, Sevilla, Spain, pages 214–228.
- Joseph Johnson. 2021. *Most common languages used on the internet as of january 2020, by share of internet users*. <https://www.statista.com/statistics/262946/share-of-the-most-common-languages-on-the-internet/>. Accessed: 2021-02-23.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Man-

- dar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. [Hatexplain: A benchmark dataset for explainable hate speech detection](#). *CoRR*, abs/2012.10289.
- Ye Nan, Kian Chai, Wee Lee, and Hai Leong Chieu. 2012. [Optimizing f-measure: A tale of two approaches](#). *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, 1:1–8.
- D. Opitz and R. Maclin. 1999. [Popular ensemble methods: An empirical study](#). *Journal of Artificial Intelligence Research*, 11:169—198.
- Òscar Garibo i Orts. 2019. [Multilingual detection of hate speech against immigrants and women in Twitter at SemEval-2019 task 5: Frequency analysis interpolation for hate in speech detection](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 460–463, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135, Copenhagen, Denmark. Association for Computational Linguistics.
- S. Sood, J. Antin, and E. Churchill. 2012a. [Profanity use in online communities](#). *Conference on Human Factors in Computing Systems - Proceedings*, pages 1481–1490.
- S. Sood, J. Antin, and E. Churchill. 2012b. Using crowdsourcing to improve profanity detection. In *Wisdom of the Crowd - Papers from the AAAI Spring Symposium*, AAAI Spring Symposium - Technical Report, pages 69–74. 2012 AAAI Spring Symposium ; Conference date: 26-03-2012 Through 28-03-2012.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *CoRR*, abs/1910.03771.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. [Ex machina: Personal attacks seen at scale](#).
- Dawei Yin, Zhenzhen Xue, Liangjie Hong, Brian Davison, April Edwards, and Lynne Edwards. 2009. [Detection of harassment on web 2.0](#). In *Proceedings of the Content Analysis in the WEB 2.0 (CAW2.0) Workshop at WWW2009*, pages 1–7.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffenseEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

## **A Our models results within SemEval-2021 Task 5 scores**

Figure 1 presents the outcome of SemEval-2021 Task 5 contest until the deadline and the green dot shows our best result at the time of publication.

## **B Original Dataset Inconsistency**

Firstly, there are words annotated as toxic in some comments, while in the other ones they are left out, despite the similar context of the utterance. The words “stupidity” or “crooked” can serve as an examples, sometimes being omitted, sometimes being treated as full toxic spans and finally, sometimes being treated as parts of toxic spans, together with their modifiers (see Table 5).

Another issue related to inconsistency is the length of the annotated span. The majority of spans consist of one to three words, but there are also cases in which spans are longer, containing not just a toxic word, but also a longer phrase including the toxic word (see Table 6). As previously, in our opinion the discrepancies are not justified by context.

Other issues we found peculiar in the provided annotation include annotating non-toxic words while omitting toxic ones (see Table 7) and beginning or ending the annotation in the middle of a word (Table 8). Such cases do not appear as often as the aforementioned discrepancies, but are also present.

Everything mentioned above might have been introduced to the dataset on purpose, as noise, in order to make the task more challenging to the models. However, we found the scale of the inconsistencies particular, the more so as it was not mentioned in the instructions for contest participants.



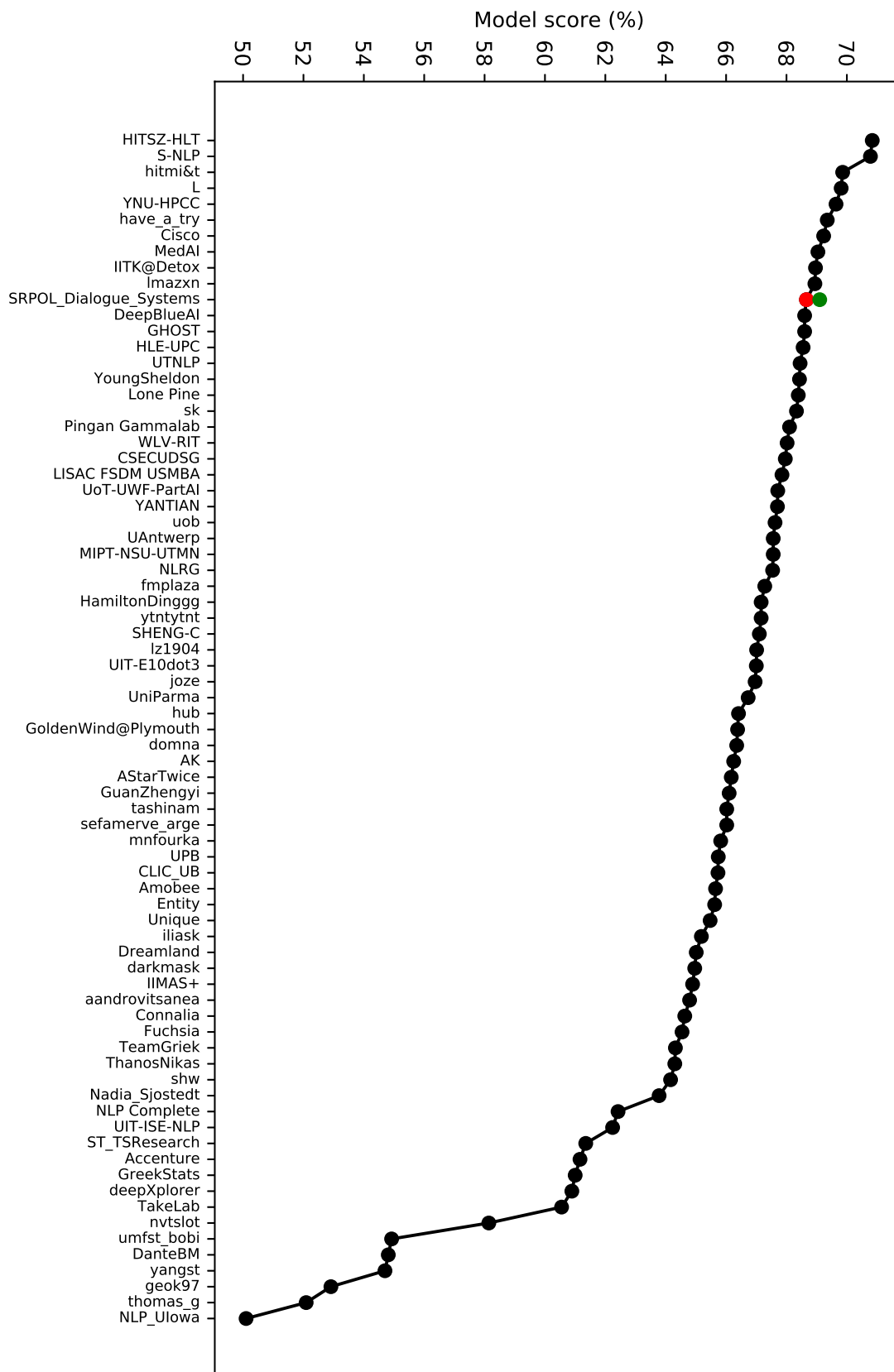


Figure 1: SemEval-2021 Task 5 results for models with scores  $\geq 50\%$  on the test dataset. The red dot points our model outcome until the deadline (F1=68.65%) and the green dot shows our best result at the time of publication (F1=69.09%).

Comment text (annotated spans in bold italics)
we dont need to move from anything other than left wing stupidity like yours (no annotation)
keep living in your left wing <i>stupidity</i> , there will be a market for our OIL from the SANDS for generations
oh more <i>left wing stupidity</i> ... colleges need to focus on educating and leave policy to the grown ups, like trump who is not banning anyone but suspending things
Unfortunately these people refuse to stand up to oppose the crooked self and friends rulers who don't hesitate to kill to kill them. (no annotation)
They were just <i>crooked</i> liars looking out for their own interests. Trump is way beyond that.
Now we will have endless <i>crooked honkey</i> lawyers talking unsuspecting "useful <i>idiots</i> " into filing outrageous claims.

Table 5: Examples of inconsistent annotation.

Comment text (annotated spans in bold italics)
I am watching Goodale on TV giving statements about this. <i>He is such an ass.</i> (17)
Saw him on the news. What an arrogant <i>arse!</i> (4)
<i>Trump's a liar</i> (14); <i>Trump's lawyer is a liar</i> (23). Get them outta there! Lock them up!
There is no indication that Trump will change from the <i>racist</i> (6), <i>sexist</i> (6), <i>demagogue</i> (9) that he is.
<i>The dotard almost forgot to sign it until his minder pulled him back</i> (68).
<i>Who gives a flying freak about what Kim and Park are making or not making? Really? This is what the SA chooses to write editorials about</i> (136)?
Trump simply does not care. "I am president, and to <i>hell</i> with all of you".

Table 6: Examples of inconsistency in regard to span length. The numbers in brackets indicate the number of characters within a given toxic span.

Comment text (annotated spans in bold italics)
I am totally with you on this continued stupidity that has been US policy, and now Hawaii's effort. I was a military officers wife expecting my second son, stationed in Germany at this time. As I recall, Kennedy gave Kruschev an opportunity to "save face". I doubt that Trump has the intelligence or skill, nor has he picked an intelligent UN rep. Or cabinet. This is where investigative journalism makes the difference. I am glad that Civil Beat is <i>challenging</i> the status quo mentality.
<i>Patriotism</i> : the last refuge of scoundrels..
Is this important news? Is it even <i>news</i> ? Who needs to know this <i>crap</i> ?
Yeah John, you tell all those <i>Chinese</i> and <i>Muslims</i> you hate to get the hell out of Canada.

Table 7: Examples of annotation of non-toxic words while omitting potentially toxic ones.

Comment text (annotated spans in bold italics)
It's so weird to have an actual <i>stupid narcissistic</i> childish person in the white house.
Yep you are <i>a sore loser</i>
The <i>hypocrisy</i> of the christian right is astounding.

Table 8: Examples of annotations starting or ending in the middle of a word.

Original annotation	Our re-annotation
“Watch people die from taking away their healthcare” DING DING DING! Winner of <i>stupid</i> post of the day award!	“ <i>Watch people die</i> from taking away their healthcare” DING DING DING! <i>Winner of stupid post of the day award!</i>
They’re not patriots. They’re vandals, <i>thieves</i> , and <i>bullies</i> . They’ve plastered a facade of patriotism over their outrage at being expected to obey the law.	They’re not patriots. <i>They’re vandals, thieves, and bullies.</i> They’ve plastered a facade of patriotism over their outrage at being expected to obey the law.
Rabidly <i>anti-Canadian</i> troll.	<i>Rabidly anti-Canadian troll.</i>

Table 9: Examples of differences between the original annotation and our re-annotation. The annotated spans are in bold italics.

## C Re-annotation

Because of inconsistencies in the original annotation, we decided to re-annotate the trial and test datasets and compare the differences. For this task we invited a team of 14 language experts (all of them with a master’s degree in linguistics). They received the same instruction as the one provided by the contest organizers. Each text was evaluated by three language experts, as in the original dataset creation process.

The differences between the original annotation and the one provided by our language experts are quite noticeable. Firstly, we believe our annotation is more coherent and involves fewer mistakes described earlier. Moreover, our language experts marked fragments as toxic more often: whole sentences, paragraphs or even whole comments, as well as more single words (see Table 9). The goal of our re-annotation was to evaluate the quality of the original datasets annotation and check if our understanding of toxicity is equivalent to the one of contest organizers.

# SINAI at SemEval-2021 Task 5: Combining Embeddings in a BiLSTM-CRF model for Toxic Spans Detection

Flor Miriam Plaza-del-Arco, Pilar López-Úbeda  
L. Alfonso Ureña-López, M. Teresa Martín-Valdivia

Department of Computer Science, Advanced Studies Center in ICT (CEATIC)  
Universidad de Jaén, Campus Las Lagunillas, 23071, Jaén, Spain  
{fmplaza, plubeda, laurena, maite}@ujaen.es

## Abstract

This paper describes the participation of SINAI team at Task 5: Toxic Spans Detection which consists of identifying spans that make a text toxic. Although several resources and systems have been developed so far in the context of offensive language, both annotation and tasks have mainly focused on classifying whether a text is offensive or not. However, detecting toxic spans is crucial to identify why a text is toxic and can assist human moderators to locate this type of content on social media. In order to accomplish the task, we follow a deep learning-based approach using a Bidirectional variant of a Long Short Term Memory network along with a stacked Conditional Random Field decoding layer (BiLSTM-CRF). Specifically, we test the performance of the combination of different pre-trained word embeddings for recognizing toxic entities in text. The results show that the combination of word embeddings helps in detecting offensive content. Our team ranks 29th out of 91 participants.

## 1 Introduction

The advance of online communication has increased the use of offensive or toxic language in several websites, including social networks such as Instagram, Twitter, or YouTube. Consequently, this type of prejudiced communication could lead to negative psychological effects among Internet users, causing anxiety, harassment, and even suicide in extreme cases (Hinduja and Patchin, 2010).

Moderation is essential to promote healthy online communication. Therefore, governments, online communities, and social media platforms are continuously taking appropriate actions to implement laws and policies combating toxic language on the Web. In order to help to track this type of comments and due to the amount of data generated every day on the Web, automatic systems based

on Natural Language Processing (NLP) techniques are required. In particular, offensive language detection and analysis has become an important area of research in NLP, resulting in several studies that are contributing to combating this website phenomenon (Plaza-del Arco et al., 2019; Zampieri et al., 2019a; Ranasinghe et al., 2019; Plaza del Arco et al., 2020).

In this paper<sup>1</sup>, we present our proposal system as part of our participation in SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021), which aims to identify entities that refer to a toxic language in the text. To accomplish the task, our team focused on detecting specific types of toxic entities in the text using a methodology based on the BiLSTM-CRF model showing that the combination of different pre-trained language embeddings succeeds in detecting toxic entities.

The rest of the paper is structured as follows. In Section 2 some previous related studies are introduced. In Section 3 we explain the data used in our methods and we describe the architecture of our proposed system to the Toxic Spans Detection task. In Section 4 we discuss the analysis and evaluation results for the experiments we performed. Finally, we conclude in Section 5 with remarks and future work.

## 2 Related work

Heretofore, several shared tasks have been organized in the NLP field to detect offensiveness on the Web for different languages. For instance, the well-known offensive language task OffensEval has held two editions in the International Workshop on Semantic Evaluation (SemEval) (Zampieri et al., 2019b,a) introducing as the main novelty in the second edition a multilingual dataset comprising 5

<sup>1</sup>NOTE: This paper contains examples of potentially explicit or offensive content which may be offensive to some readers. They do not represent the views of the authors.

languages. The GermEval shared task focused on the identification of offensive language in German tweets and comprised two tasks, a coarse-grained binary classification task and a fine-grained multi-class classification task (Wiegand and Siegel, 2018). For Spanish, as far as we know, the first task on offensive language appeared at the 3rd SEPLN Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval) (Carmona et al., 2018) whose goal was to detect aggressiveness Mexican Spanish Tweets.

As a result, most of the studies and resources in offensive language research have been developed specifically for binary and multi-class classification tasks (Ranasinghe et al., 2019; Plaza-del Arco et al., 2019, 2020). However, other tasks such as Named Entity Recognition (NER) play an important role in this research and are essential to identify the entities that make a text toxic. Highlighting these toxic spans can help human moderators to interpret and identify easily this type of content on the Web instead of relying on a system that generates a score of unexplained toxicity per post. NER aims to identify and classify named entities mentioned in unstructured text into predefined categories. The earliest systems developed for addressing this task did not use training data but worked based on handcrafted features, heuristics, and a set of rules (Nadeau and Sekine, 2007; Collins and Singer, 1999; López-Ubeda et al., 2018). However, the cost of manual feature tagging and the poor obtained results lead to deep learning-based techniques as the most suitable choice to tackle the task by discovering patterns and learning the features in an end-to-end manner (López-Úbeda et al., 2020). Existing state-of-the-art approaches for sequence labeling have proven that Recurrent Neural Networks (RNNs) are capable of learning useful representations automatically as they enable the modeling of long-distance dependencies between words in a sentence (Limsopatham and Collier, 2016; Wintaka et al., 2019). Inspired by these studies, we have developed a system based on BiLSTM-CRF model along with the combination of different types of word embeddings to address the toxic spans detection task in text.

### 3 Named Entity Recognition Methodology

To address the toxic detection task, we focus on recognizing and extracting specific types of toxic enti-

ties in the text. Specifically, we follow a methodology proposed by (Huang et al., 2015) implementing a BiLSTM-CRF model for the NER task.

#### 3.1 Word Embeddings

As input layer of the BiLSTM-CRF neural network we have combined the following word embeddings:

- **Static Word Embedding.** We use GloVe embeddings which are static and word-level, i.e. each distinct word gets exactly one pre-computed embedding. This type of embeddings is context-independent (Pennington et al., 2014).
- **Contextual Word Embedding.** For our experiments, we tested two different contextual pre-trained word embeddings: BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018) and XLM (Lample and Conneau, 2019). Unlike the previous ones, they are context-dependent which means they produce word representations that are dynamically informed by the words around them. They are based on the well-known Transformer (Vaswani et al., 2017), an attention mechanism that processes the entire text input simultaneously to learn contextual relations between words (or sub-words). Specifically, we used the *xlm-mlm-en-2048* model and the *bert-base-cased* model provided in HuggingFace (Wolf et al., 2019).

#### 3.2 BiLSTM-CRF architecture

We use the combination of bidirectional LSTM and CRF to identify the toxic spans. The context of each word in the sentence is captured by the BiLSTM and then the predictions on the entities are simultaneously performed in the CRF layer (Sutton and McCallum, 2006). The architecture of BiLSTM-CRF model is illustrated in Figure 1. This architecture follows a sequence of layers as follows:

- Embedding layer. Each word of the sentence is mapped to a vector of concatenated embeddings. As mentioned above, in our experiments, we use XLM, BERT, and GLOVE embeddings.
- BiLSTM layer. A bidirectional LSTM recurrent network takes as input the embeddings. In sequence tagging tasks, for a specific time

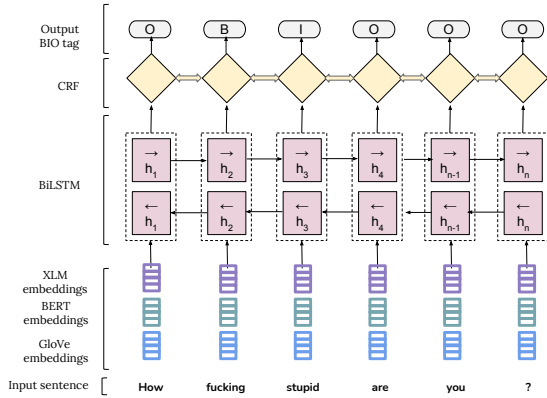


Figure 1: Proposed system architecture based on a BiLSTM-CRF neural network.

frame, this layer enables the hidden states to capture both historical and future context information and then to label a token.

- **CRF layer.** It allows to use efficiently historical and future tags to predict the current tag.

## 4 Data and Experimental Setup

### 4.1 Dataset preprocessing

We use the English dataset provided by the organizers in SemEval 2020 Task 5: Toxic Spans Detection. The dataset is split into three different subsets: train, trial, and test, consisting of 7,939, 690, and 2,000 instances, respectively.

Each instance in the dataset comprises two fields, the text and a list of toxic spans. A toxic span is defined as a sequence of characters in words that attribute to the text’s toxicity. If the text does not contain toxic spans, the span list is empty. An example of two instances in the dataset is provided in Table 1. In the first example, the word “crap” is labeled as toxic in the text, which has character offsets from 15 to 18. The second example includes the toxic span “idiot” which has character offsets from 4 to 8.

Text	Spans
What a load of <b>crap</b> .	[15, 16, 17, 18]
You <b>idiot</b> . The media went to war against truth.	[4, 5, 6, 7, 8]

Table 1: Two instances in the dataset. Toxic words are highlighted in the text.

To perform our experiments, we preprocess the subsets of the dataset in the following way. First,

we used the `nltk.tokenize` package<sup>2</sup> to tokenize the text. Then, we generated the following features for each text in the subset: the word, the position of the beginning and end of the word in the text, and the NER tag. In order to perform the NER tagging, we follow the BIO annotation scheme to label multi-token named entities (Ratinov and Roth, 2009), which represents that the label is the beginning of a span (B-Toxic), inside the span (I-Toxic), or belongs to no span (O). This scheme is the most popular in the NER task. Figure 2 shows an example of the features generated for the following example in the training set: “How fucking stupid are you?”, spans: [4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17].

How	0	3	O
fucking	4	11	B-Toxic
stupid	12	18	I-Toxic
are	19	22	O
you	23	26	O
?	26	27	O

Figure 2: Example of training set instance with generated features using BIO annotation scheme.

### 4.2 Experiments

During the pre-evaluation period, we trained our models on the train set and evaluated our different approaches on the trial set. During the evaluation period, we trained our models on the train and trial sets and tested the model on the test set.

Flair’s framework (Akbik et al., 2019) builds directly on Pytorch was used to design the BiLSTM-CRF network. We used the default hyperparameter setting in Flair with the following configuration: learning rate as 0.1, batch size as 32, dropout probability as 0.01, and maximum epoch as 300. All experiments (training and evaluation) were performed on a node equipped with two Intel Xeon Silver 4208 CPU at 2.10GHz, 192GB RAM, as main processors, and six GPUs NVIDIA GeForce RTX 2080Ti (with 11GB each).

Our team (SINAI) submitted 4 runs for the Toxic Spans Detection task and each run evaluates the word embeddings as an input to the BiLSTM-CRF network, as explained in Section 3.

#### Run 1. GloVe embeddings.

<sup>2</sup><https://www.nltk.org/api/nltk.tokenize.html>

**Run 2.** BERT embeddings.

**Run 3.** XLM embeddings.

**Run 4.** BERT + XLM + GloVe embeddings.

## 5 Results

In this section, we present the results obtained by our proposed system. In order to evaluate them, we use the official competition metric F1-score.

The results of our participation in the Toxic Spans Detection task during the evaluation phase are shown in Table 2. In particular, we list the performance of the four runs submitted using the BiLSTM-CRF model along with the combination of different word embeddings. If we analyze the results of the first 3 runs (each embeddings independently), we notice that they slightly differ, the best result is achieved by the contextual embedding XLM. However, training the model on the combination of static and contextual embeddings (GloVe, BERT, and XLM) leads to enhanced performance with a 0.6727 F1-score. Therefore, our results show the success of the combination of embeddings we chose to solve the task of toxic spans detection in comments using the proposed model.

Run	Word embeddings	F1-score
1	GloVe	0.6618
2	BERT	0.6606
3	XLM	0.6635
<b>4</b>	<b>BERT + XLM + GloVe</b>	<b>0.6727</b>

Table 2: Systems test results achieved by SINAI in SemEval Task 5: Toxic Spans Detection.

Table 3 shows the official rank in the competition. As we can see, we are ranked 29th out of 91 participating teams obtaining an F1-score of 0.6727 with our system. The best result was obtained by the team HITSZ-HLT with an F1-score of 0.7083, which differs from our results achieved by 3.56%. In general, low results for the task are obtained which shows the Toxic Spans Detection as a challenge to be addressed by the NLP community and, therefore, further research is needed to advance on this specific task. We also observe that the number of participants in this task is high (91) which shows the importance and interest of the NLP community in contributing to addressing this challenge.

User name (ranking)	F1-score
HITSZ-HLT (1)	0.7083
lmazxn (10)	0.6893
<b>SINAI (29)</b>	<b>0.6727</b>
UIT-ISE-NLP (63)	0.6223
ramya.akula01 (85)	0.1968
AmrHendy (91)	0.0675

Table 3: System Results per participating team in Task 5: Toxic Spans Detection.

## 6 Conclusions and Future Work

This paper presents the participation of the SINAI research group in Task 5: Toxic Spans Detection at SemEval 2021.

In this paper, we use a deep learning-based approach for NER to identify spans that make a text toxic, which focuses on the use of a BiLSTM-CRF neural network where different word embeddings are tested. The model is trained on the dataset provided by the organizers of the task (Pavlopoulos et al., 2021) and preprocessing techniques are carried out to tokenize and tagged the dataset by using the BIO scheme.

Our results show that the sophisticated BiLSTM-CRF architecture which has been successfully used for other tasks such as biomedical entity recognition or part-of-speech tagging, but also achieves remarkable results when addressing tasks related to the identification of offensive language in comments. Besides, we find that this architecture with our proposed combination of embeddings for word representation provides useful insights for the learning phase of the neural network achieving better results than training the network with a single type of word embedding.

For future work, we plan to study the performance of our proposed method using a variety of linguistic features, including emotions that are inextricably linked to offensive language.

## Acknowledgments

This work has been partially supported by a grant from the Ministry of Science, Innovation, and Universities (Scholarship FPI-PRE2019-089310), Fondo Europeo de Desarrollo Regional (FEDER), and LIVING-LANG project (RTI2018-094653-B-C21) from the Spanish Government.

## References

- Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. Flair: An easy-to-use framework for state-of-the-art nlp. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Flor Miriam Plaza-del Arco, M. Dolores Molina-González, Maite Martin, and L. Alfonso Ureña-López. 2019. SINAI at SemEval-2019 task 6: Incorporating lexicon knowledge into SVM learning to identify and categorize offensive language in social media. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 735–738, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Flor Miriam Plaza del Arco, M. Dolores Molina González, Alfonso Ureña-López, and Maite Martin. 2020. SINAI at SemEval-2020 task 12: Offensive language identification exploring transfer learning models. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1622–1627, Barcelona (online). International Committee for Computational Linguistics.
- Flor Miriam Plaza-del Arco, M Dolores Molina-González, L Alfonso Ureña-López, and M Teresa Martín-Valdivia. 2020. Comparing pre-trained language models for Spanish hate speech detection. *Expert Systems with Applications*, 166:114120.
- Miguel Ángel Álvarez Carmona, Estefanía Guzmán-Falcón, Manuel Montes-y-Gómez, Hugo Jair Escalante, Luis Villaseñor Pineda, Verónica Reyes-Meza, and Antonio Rico Sulayes. 2018. Overview of MEX-A3T at IberEval 2018: Authorship and Aggressiveness Analysis in Mexican Spanish Tweets. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018), Sevilla, Spain, September 18th, 2018*, volume 2150 of *CEUR Workshop Proceedings*, pages 74–96. CEUR-WS.org.
- Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Sameer Hinduja and Justin W Patchin. 2010. Bullying, cyberbullying, and suicide. *Archives of suicide research*, 14(3):206–221.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*.
- Nut Limsopatham and Nigel Collier. 2016. Bidirectional LSTM for Named Entity Recognition in Twitter Messages.
- Pilar López-Ubeda, Manuel C Díaz-Galiano, María Teresa Martín-Valdivia, and L Alfonso Ureña-López. 2018. SINAI en TASS 2018 Task 3. Clasificando acciones y conceptos con UMLS en MedLine. *Proceedings of TASS*, 2172.
- Pilar López-Úbeda, MC Díaz-Galiano, MT Martín-Valdivia, and L Alfonso Ureña-López. 2020. Extracting Neoplasms Morphology Mentions in Spanish Clinical Cases through Word Embeddings. *Proceedings of IberLEF*.
- David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection. In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Tharindu Ranasinghe, Marcos Zampieri, and Hansi Hettiarachchi. 2019. BRUMS at HASOC 2019: Deep Learning Models for Multilingual Hate Speech and Offensive Language Identification. In *FIRE (Working Notes)*, pages 199–207.
- Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.
- Charles Sutton and Andrew McCallum. 2006. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, 2:93–128.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Michael Wiegand and Melanie Siegel. 2018. Overview of the GermEval 2018 Shared Task on the Identification of Offensive Language. In *Proceedings of KONVENS 2018*.



Deni Cahya Wintaka, Moch Arif Bijaksana, and Ibnu Asror. 2019. Named-Entity Recognition on Indonesian Tweets using Bidirectional LSTM-CRF. *Procedia Computer Science*, 157:221–228.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019a. Predicting the type and target of offensive posts in social media. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019b. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

# CSECU-DSG at SemEval-2021 Task 5: Leveraging Ensemble of Sequence Tagging Models for Toxic Spans Detection

Tashin Hossain, Jannatun Naim, Fareen Tasneem,  
Radiathun Tasnia, and Abu Nowshed Chy

Department of Computer Science and Engineering  
University of Chittagong, Chattogram-4331, Bangladesh

{tashin.hossain.cu, jannatun.naim.cu, fareen.tasneem,  
radia.tasnia.cu}@gmail.com and nowshed@cu.ac.bd

## Abstract

The upsurge of prolific blogging and microblogging platforms enabled the abusers to spread negativity and threats greater than ever. Detecting the toxic portions substantially aids to moderate or exclude the abusive parts for maintaining sound online platforms. This paper describes our participation in the SemEval 2021 toxic span detection task. The task requires detecting spans that convey toxic remarks from the given text. We explore an ensemble of sequence labeling models including the BiLSTM-CRF, spaCy NER model with custom toxic tags, and fine-tuned BERT model to identify the toxic spans. Finally, a majority voting based fusion method is used to determine the unified toxic spans. Experimental results depict the competitive performance of our model among the participants.

## 1 Introduction

Social media being a key factor in the world dynamics and toxicity in user-generated contents is a real threat. Threats and hatred instigated in posts and blogs implants fear in users' minds and prevents them from sharing their creative thoughts, valuable opinions to critical information. Sometimes it leads to severe mental trauma and fatalities. Hence, it is a formidable task to precisely detect toxicity in comments and posts to be able to moderate those portions and provide the users a safe online platform to express themselves.

Toxic span detection is a process where the specific toxic segment of a text is detected instead of detecting the whole text as toxic. The goal of this task is to eradicate the vagueness that is present in simple toxic text classification models and help the moderator to precisely moderate the toxic portions instead of the whole post. To elucidate the task, two examples are presented in Table 1.

The first four authors have equal contributions.

---

**Text#1:** How fucking stupid are you?  
**Span:** [4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17]

---

**Text#2:** What a sociopathic and parasitic leader we have.  
**Span:** [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 23, 24, 25, 26, 27, 28, 29, 30, 31]

---

Table 1: Example of sample texts with toxic spans.

Here, the “fucking stupid” portion of Text#1 is toxic and is attacking the personality of the second person, so the indices of this portion are included in the toxic span. In Text#2, the “sociopathic and parasitic” fragment is used as a toxic adjective to describe the leader in that context. Consequently, the indices of this fragment are incorporated in the span. We need to detect such spans accurately to remove toxicity from user content and preserve the safe and sound flow of online information.

Toxic content detection on online platforms is a state-of-the-art notion. Numerous works have been done on the binary and multi-label classification of toxic texts. For instance, Georgakopoulos et al. (Georgakopoulos et al., 2018) investigated the impact of CNN in toxic comment classification against the traditional bag-of-words approaches. A multiple word embedding-based approach was adopted by Carta et al. (Carta et al., 2019) for multi-class multi-label toxic comment classification. Besides, the effectiveness of feature extraction in hate speech detection was explored by Schmidt et al. (Schmidt and Wiegand, 2017). Multitude of datasets on toxic comments such as dataset based on Wikipedia discussion comments (Wulczyn et al., 2017), comments on online forums (Borkan et al., 2019a), and offensive language identification dataset (OLID) (Zampieri et al., 2019) were also introduced.

However, very few works detect the precise toxic span from text contents. Katsiolis et al. (Katsiolis, 2020) experimented on both unsupervised and su-

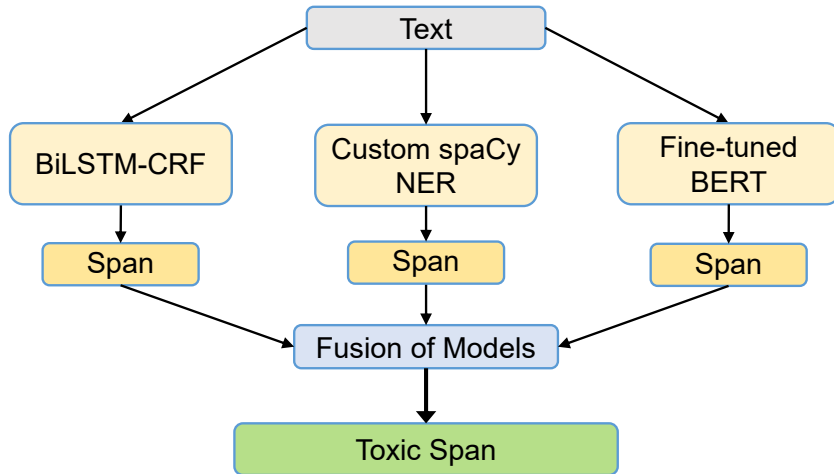


Figure 1: Overview of our proposed framework.

pervised methods to address this challenge. The unsupervised methods include the input erasure method and the LIME algorithm whereas the supervised method implements sequence labeling through a BERT model. The unintended bias created in publicly used toxicity detection models due to many reasons such as the influence of regional culture was investigated by Borkan et al. (Borkan et al., 2019a). John et al. (Pavlopoulos et al., 2017) surveyed the impact of user embeddings, user type embeddings, user biases, or user type biases on the RNN-based moderation method.

In this paper, we portray our insights acquired from experimenting on this task. We propose an approach focusing on an ensemble of sequence labeling models including the BiLSTM-CRF, spaCy NER model with custom toxic tags, and fine-tuned BERT model. We procure the spans from these models through a majority voting scheme to determine the final toxic spans.

The organization of this paper is as follows: we elucidate our proposed framework in Section 2. Section 3 encompasses the experimental details and comparative performance analysis. Finally, we conclude this paper with some future notions in Section 5.

## 2 Proposed Framework

We cast the toxic span detection as a sequence tagging task and employ an ensemble of sequence tagging models. Our proposed system comprises three individual models. The framework of our system is depicted in Figure 1. The first model is a BiLSTM-CRF model with the BIO tagging scheme. The second model is a custom spaCy named entity

recognition (NER) model. The third model is a fine-tuned BERT model for token classification. We leverage these three models as sequence tagging models. These models generate tags in token level for a text. Subsequently, we extract span based on the toxic tags. Finally, we apply a majority voting based fusion scheme on these spans and determine the final toxic spans.

### 2.1 BiLSTM-CRF

The BiLSTM-CRF model is well-known for sequence-tagging tasks such as named entity recognition (NER). We utilize the model implemented by (Reimers and Gurevych, 2017). For training purposes, the dataset needs to be in CoNLL-2003<sup>2</sup> format where two columns for tokens and BIO tags are required. Since it requires the text to be in a tokenized form, we tokenize the text using NLTK TweetTokenizer (Bird et al., 2009). After tokenization, we label the tokens with custom tags such as B-TOX(begin), I-TOX(inside), and O(outside) utilizing the toxic span from the training dataset. These tokens are then sent to the embedding layer. The embedding layer has three variants of embeddings: word embedding, casing feature or capitalization feature, and character embedding. We employ pre-trained GloVe (6B) (Pennington et al., 2014) word embedding and a CNN based character embeddings (Ma and Hovy, 2016). The embedding vectors are concatenated and the output is fed to the BiLSTM encoder which tags tokens with the BIO tagging scheme. The BiLSTM encoder is followed by a CRF classifier where the tags are optimized enforcing the intermediate logic of tags.

<sup>2</sup><https://www.clips.uantwerpen.be/conll2003/ner/>

## 2.2 Custom spaCy NER

We exploit the spaCy (Honnibal and Montani, 2017) to build an NER type sequence labeling model with the custom tag “TOXIC”. We convert the dataset to spaCy entity format and load a spaCy blank English model. We append new word vectors utilizing a pre-trained word2vec (Mikolov et al., 2013) model. Consequently, we add NER pipeline to the model and also a “TOXIC” label. We disable all the pipelines except NER and loop through the training dataset several times.

## 2.3 Fine-tuned BERT

We finetune the state-of-the-art bert-large-cased model (Devlin et al., 2019) to identify the toxic spans. We employ the BertForTokenClassification (Wolf et al., 2019) method to perform the token level tagging. This method classifies level for each tokenized word in a sentence. To generate the training data, we convert the sentence into tokens and annotate them with spans. We tag the tokens as “non-toxic” and “toxic” whereas the tokens that are tagged as “toxic” are in between the spans.

## 2.4 Fusion of Models

An ensemble approach is a simulation that constructs multiple models and then blends them to bring out improved results. To obtain a more accurate solution than a single model, we apply majority voting (Rokach, 2010) on the spans generated from three models as shown in Figure 1. The primary idea is based on the frequency of the span elements. If a span is predicted by at least two models, it is included in the final predicted span. Thus, we obtain our final toxic spans through majority voting.

# 3 Experiments and Evaluations

## 3.1 Dataset Description

For detecting toxic spans in posts, we used the Civil Comments Dataset (Borkan et al., 2019b) which consists of 10K toxic comments. The whole dataset is divided into three subsets where the train, trial, and test set comprises 7939, 690, and 2000 comments, respectively. Toxic comments are mainly divided into two portions: 1. Having no toxic spans and 2. Having toxic spans that are identified as spans with specific character positions. Analyzing the ratio of empty and toxic spans in our dataset we found that 90% of data occupies toxic spans where only 10% data have empty spans. F1-Score is used as the primary evaluation metric in this task.

## 3.2 Experimental Setup

In our CSECU-DSG system submitted to the SemEval-2021 Task 5 (Pavlopoulos et al., 2021), we make use of three sequence and entity tagging models to get better predictions. We present the configuration of our best submitted system in Table 2. Based on the predicted spans from these models, a majority voting has been applied.

System	Settings
BiLSTM-CRF	<ol style="list-style-type: none"><li>1. <i>dropout</i>: (0.25, 0.25)</li><li>2. <i>LSTM-Size</i>: [100, 100]</li><li>3. <i>maxCharLength</i>: 50</li><li>4. <i>Tokenizer</i>: TweetTokenizer</li><li>5. <i>Word embedding</i>: GloVe (6B)</li><li>6. <i>Optimizer</i>: nadam</li><li>7. <i>miniBatchSize</i>: 32</li><li>8. <i>Epochs</i>: 25</li></ol>
Custom spaCy NER	<ol style="list-style-type: none"><li>1. <i>spaCy Model</i>: blank (‘en’)</li><li>2. <i>Pipeline</i>: ner</li><li>3. <i>Word embedding</i>: word2vec</li><li>4. <i>Iteration</i>: 30</li><li>5. <i>drop</i>: 0.5</li></ol>
Fine-tuned BERT Model	<ol style="list-style-type: none"><li>1. <i>Tokenizer</i>: bert-large-cased</li><li>2. <i>Optimizer</i>: AdamW</li><li>3. <i>Batch Size</i>: 16</li><li>4. <i>Learning_rate</i>: 2e-5</li><li>5. <i>weight_decay_rate</i>: 0.01</li><li>6. <i>Epochs</i>: 25</li></ol>

Table 2: System settings.

## 3.3 Results Analysis

Now, we compare the performance of our system against other competitors’ systems. Among the 91 valid submissions, the comparative performance with top-performing teams depicted in Table 3.

Team Name	F1-Score
HITSZ-HLT9 (1st)	0.7083028253
hitmi&t (3rd)	0.6984762534
IITK@Detox (9th)	0.6895352367
<b>CSECUDSG (21st)</b>	<b>0.6795264755</b>
mnfourka (45th)	0.6581458018
ST_TSRResearch (64th)	0.6133591537

Table 3: Comparative performance analysis.

It depicts that our system achieved competitive performance compared to the participants’ systems. It only lacks by 3% from the top-performing team HITSZ-HLT.

## 4 Discussion

To estimate the impact of individual components on the overall system’s performance, we examine the performance of individual models on the test set. To do this, we make use of the test set and the findings are presented in Table 4.

Method	F1-Score
CSECU-DSG	0.6795264755
Performance of Individual Model	
–Fine-tuned BERT	0.6381618923
–Custom spaCy NER	0.6474175682
–BiLSTM-CRF	0.6404340000

Table 4: Performance analysis of individual models.

It shows that all three models obtained a similar kind of performance. However, employing the majority voting based scheme on these three models improves the overall result by almost 3% which leads to better detection of toxic spans from the text. Thus, we demonstrate the efficacy of utilizing the ensemble strategy to ameliorate the performance.

To qualitatively demonstrate the effectiveness of the ensemble approach compared to the individual models an instance is illustrated in Table 5. It clearly shows that majority voting helps to detect the accurate span.

<b>Text:</b> They are more animal than the goat, disgusting!!!!
<b>Gold:</b> [36, 37, 38, 39, 40, 41, 42, 43, 44, 45]
<b>BiLSTM-CRF:</b> [30, 31, 32, 33, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45]
<b>Custom spaCy NER:</b> [36, 37, 38, 39, 40, 41, 42, 43, 44, 45]
<b>Fine-tuned BERT:</b> [14, 15, 16, 17, 18, 19, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45]
<b>Majority voting:</b> [36, 37, 38, 39, 40, 41, 42, 43, 44, 45]

Table 5: Comparative performance analysis of models according to the predicted toxic spans.

We further investigate the reason behind the erroneous span detection by our proposed system. In this regard, we articulate some examples in Table 6.

Text	Predicted Span	Gold Span
1. See a shrink you pathetic troll.	[17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 28, 29, 30]	[17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]
2. ADN is endorsing, without officially endorsing. Bunch of cowards !!!	[58, 59, 60, 61, 62, 63, 64]	[49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64]
3. The mascot was a ridiculous pick twenty years ago, too. Did you ever see the welcome sign going into Keenesburg? "Home to 500 people and a few sore-heads."	[17, 18, 19, 20, 21, 22, 23, 24, 25, 26]	17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31]

Table 6: Examples of erroneous span detection.

We observed that our system could not detect the in-between spaces of toxic words. Such as in the first example, the predicted span is “pathetic” (17-24) and “troll” (26-30). Whereas, the gold span is “pathetic troll”(17-30). The probable reason for this can be that our models are trained with tokens of the training dataset. Another observation indicates that our system failed to detect the phrasal spans of some texts. In example #2 and #3, we see that instead of capturing the toxic phrases “Bunch of cowards” and “ridiculous pick”, it detects the toxic words only. Since two of our models are trained on token-level and only the BiLSTM-CRF model follows the BIO tags convention, the ensemble of models lacks in perceiving the context of the phrasal toxic texts and sometimes fragments the toxic sequences. Though majority voting improves the overall score, it shrinks some important features of the discrete models.

## 5 Conclusion and Future Directions

In this paper, we introduced an ensemble of three distinct models to detect the toxic spans. Among these models, BiLSTM-CRF and Custom spaCy NER models are implemented as NER type sequence and entity tagging models whereas fine-tuned BERT model is exploited as a token classification model. We also leveraged a majority voting strategy to overcome the limitations of individual models. Our model tackles the task challenge effectively and achieved a competitive performance compared to the participants’ systems.

Our future plan incorporates exploring a better sequence tagging model with an ensemble of various fine-tuned language models i.e. ALBERT, DistilBERT, RoBERTa, and GPT.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019a. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019b. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561.
- Salvatore Carta, Andrea Corriga, Riccardo Mulas, Diego Reforgiato Recupero, and Roberto Saia. 2019. A supervised multi-class multi-label word embeddings approach for toxic comment classification. In *KDIR*, pages 105–112.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 4171–4186.
- Spiros V Georgakopoulos, Sotiris K Tasoulis, Aristidis G Vrahatis, and Vassilis P Plagianakos. 2018. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th hellenic conference on artificial intelligence*, pages 1–6.
- Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420.
- Athanasios Katsiolis. 2020. Toxic span detection in online posts.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- John Pavlopoulos, Prodromos Malakasiotis, Juli Bakagianni, and Ion Androutsopoulos. 2017. Improved abusive comment moderation with user embeddings. *arXiv preprint arXiv:1708.03699*.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. *Glove: Global vectors for word representation*. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Nils Reimers and Iryna Gurevych. 2017. Optimal hyperparameters for deep lstm-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*.
- Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*, pages 1–10.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.
- Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. *arXiv preprint arXiv:1902.09666*.

# UTNLP at SemEval-2021 Task 5: A Comparative Analysis of Toxic Span Detection using Attention-based, Named Entity Recognition, and Ensemble Models

Alireza Salemi, Nazanin Sabri, Emad Kebriaei, Behnam Bahrak, Azadeh Shakery

School of Electrical and Computer Engineering, College of Engineering

University of Tehran, Tehran, Iran

{alireza.salemi,nazanin.sabri,emad.kebriaei,bahrak,shakery}@ut.ac.ir

## Abstract

Detecting which parts of a sentence contribute to that sentence’s toxicity—rather than providing a sentence-level verdict of hatefulness—would increase the interpretability of models and allow human moderators to better understand the outputs of the system. This paper presents our team’s, *UTNLP*, methodology and results in the SemEval-2021 shared task 5 on toxic spans detection. We test multiple models and contextual embeddings and report the best setting out of all. The experiments start with keyword-based models and are followed by attention-based, named entity-based, transformers-based, and ensemble models. Our best approach, an ensemble model, achieves an F1 of 0.684 in the competition’s evaluation phase.

## 1 Introduction

When social media platforms were first introduced, they allowed users to post content on any topic they wished, without restricting the type of content they were allowed to put out. This absence of restrictions, along with the anonymity of users through these platforms (Pinsonneault and Heppel, 1997; Mondal et al., 2017), resulted in the spread of offensive language and hate speech online. While one might think there are only a small number of users who produce these types of hateful content, it has been shown that if social media platforms are left unmoderated, over time, the language of the community as a whole will change such that it highly correlates with the speech of hateful users rather than non-hateful ones (Mathew et al., 2020). Given the huge number of social media postings every day, manual moderation of these platforms is not a possibility. As a result, many researchers began to study automatic hate speech detection. Most studies on hate speech detection only provide labels at the sentence level, showing whether

the construct as a whole is toxic or not. But these types of models, offer little explanation as to why the class was predicted, making it hard for human moderators to interpret the results (Pavlopoulos et al.).

In an attempt to solve the aforementioned issue, we took part in SemEval-2021 shared task 5 (Pavlopoulos et al., 2021), where we aim to detect which spans of a sentence cause it to become toxic. Our contributions are as follows: We begin our experimentation by evaluating a random baseline. Next, we test keyword-based methods, trying to find if toxic spans often include words that are known as hateful or negative in available word lists. We then test attention-based models, building on the hypothesis that what the attention model learns to focus on when detecting toxic speech, are the toxic spans. Afterwards, we look at the issue as a named entity recognition problem, by considering *toxic* as a named entity category. Finally, we fine tune *T5-base* and explore the possibility of looking at the task as a text-to-text problem. We compare different neural network architectures and embeddings, and report the model with the best performance. Additionally, we experiment with some hand-crafted features and evaluate their effectiveness in detecting toxic spans. Our best result, an ensemble of named-entity-based models, achieves an F1 of 0.684.

## 2 Related Work

In this section we provide a brief overview of studies on hate and toxic speech detection, followed by work on span detection in different sub-fields.

### 2.1 Hate Speech

Hate speech is defined as “any communication that disparages a person or a group on the basis of some characteristic such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other

characteristics” (Nockleby, 2000). However, no clear distinction between toxic and hateful speech has been provided in the scientific literature (D’sa et al., 2019). There are quite a few surveys on the topic of hate speech detection. (Schmidt and Wiegand, 2017), describes available methods, features, and models for such a task. Another survey conducted in 2018 (Fortuna and Nunes, 2018), offers another view of the current state of the field, as well as suggesting ways in which hate speech detection could advance further. Other surveys on the topic published in 2020 include: (Naseem et al., 2020) which examines the impact of pre-processing on the performance of hate speech models. Corpora and resources for the task are studied in (Poletto et al., 2020). Additionally, throughout the years, many shared tasks have been organized to help propel studies in the field (Vu et al., 2020; Bosco et al., 2018; Basile et al., 2019). In addition to the classification of hate speech, significant effort has been put into the analysis of the target of hate (Silva et al., 2016; ElSherief et al., 2018).

Although numerous models have been tested, (Gröndahl et al., 2018) argues that when it comes to hate speech detection, the model is less important than the labeling criteria and the type of data. In confirmation of the importance of labeling, (Arango et al., 2019) also finds that models trained on annotations done by experts outperform systems trained on amateur annotations.

## 2.2 Span Detection

Named entity recognition (NER), code-switching detection, quotation detection, and key-phrase extraction are among many tasks that involve span identification.

(Chen et al., 2020) employs SpanBERT (Joshi et al., 2020) accompanied by a sequence tagging model to detect erroneous spans, proceeding to use detected spans to perform error correction. The combination of conditional random fields (CRF) (Lafferty et al., 2001) and attention mechanisms (Vaswani et al., 2017) are explored in (Xu et al., 2020) to explore aspect sentiment classification. The study finds that the use of multiple CRFs (to some limit) does improve performance. In (Papay et al., 2020) the authors look into systems to predict the performance of span identification tasks. To do so, BIO labels are used, and it is found that BERT (Devlin et al., 2018) helps when there is little data, while CRF is of great help in hard cases. In addition, the

frequency of spans is found to help while length hurts the performance of the model. Furthermore, LSTMs (Hochreiter and Schmidhuber, 1997) are reported to require large amounts of data to learn. (Tang et al., 2019) explores using fine-tuned BERT with attention models to extract keywords, showing how such models could enable the text classification model to be human interpretable.

## 3 Data

In this section, we will provide a brief description of the datasets utilized in this study. We will begin with our main dataset in which span-level toxicity has been labeled (3.1), next we look at other datasets that were used to better train our models, namely the hate word list that was used (3.2.1) and the sentence-level hate speech data (3.2.2).

### 3.1 Main Task Dataset: Toxic Spans

The main dataset used in this study is that of the SemEval 2021, Toxic Span detection task (Pavlopoulos et al.; Borkan et al., 2019a). In this dataset, which was built upon *Civil Comments* (Borkan et al., 2019b), toxic word sequences (for sentences in the English language) have been labeled. In other words, labels are indexes of characters that are toxic in each sentence. There are a total of 8,629 sentences in the dataset, 8,101 of which include at least one annotated toxic span, and the rest have none. Sentences are on average 35 words long. The word-length of the toxic spans varies from one to 176 words. Toxic spans are, on average, 2.29 words long.

There are some issues with regard to the quality of the annotations in the dataset. Table 1 shows some examples of annotated comments in the dataset. While the first sentence is satisfactorily annotated, the second and third examples display issues with the labels in the dataset. More concretely, in the second example we can see that the indexes result in poorly separated and broken up words. Additionally, the annotated words are not toxic. In the third example we see that some of the words which do have a toxic connotation are not included in the annotation. While these examples are not extremely common in the dataset, these types of issues make automatic detection of such spans much more difficult.



Text	Toxic Spans
what load you trump chumps just do not have any idea how to deal with reality you have terrible judgment and pick exceptionally idiotic arrogant leaders trump admitted he fired comey to stop the russia investigation man is he stupid.	['idiotic', 'man is he stupid']
except for one thing they are liars they only care about being thugs	['r one th']
what harm have you ever heard of someone getting attacked by bear while taking dump in the woods please does just owning gun make someone paranoid and pu55y at the same time	['harm']

Table 1: Examples of comments and the annotated toxic spans in the dataset

## 3.2 Datasets Used for Training

To better train our models, we made use of several auxiliary datasets.

### 3.2.1 Word-list Dataset

One of the methods tested in this study is based on word-matching. In other words, we check whether each word in the sentence is among hateful words and if so predict its label to be toxic. While this method is rather simple and we acknowledge that not all hate words are toxic and they could simply be used as a joke, we consider this method as a good first step to help us better understand the task at hand. As a result we need to use a list of hate words. For that purpose, we used a list of 1,616 unique hate words found on Kaggle (nicapotato).

### 3.2.2 Hate Speech Dataset

To be able to train our attention-based models (4.1) we needed to have sentence-level annotated data. Thus we used the *Civil Comments* dataset (Jigsaw-Conversation-AI). The fact that this dataset and our main dataset have the same domain is the reason why this specific dataset was selected. In this dataset, each sentence is labeled with a number between 0 and 1, representing how hateful the text is. We consider sentences with scores above 0.5 to be hateful, and consider the rest as non-hateful. We then create a balanced sample of 289,298 sentences to train our model. The average length of sentences in this dataset is 48.12 words which is slightly longer than the sentences in the main dataset (3.1).

## 4 Methodology

In this study we have tested and compared various models to perform toxic span detection. In this section we will go over the structure and hyperparameters of these models. The codes of all models

are publicly available on GitHub<sup>1</sup>.

### 4.1 Attention-based Methods

We begin with the intuition that if a model with an attention layer is trained to detect hate speech at the sentence-level, the words the attention layer would learn to place importance on, would be the hateful words and spans. Consequently, we create a model made up of the following three layers:

- (1) BERT-Base (Uncased) Layer which encodes our input texts.
- (2) Attention Layer which is meant to be used for the aforementioned purposes
- (3) Dense Layers which connect the attention outputs to two output nodes, detecting if the text is hateful or not.

To train this model we have two training stages:

- Sentence-level classification of hate speech
- Span-level detection of toxic spans

First we perform pre-processing by removing all punctuations (except those in the middle of words such as a\$\$hole), and lower-casing all words. Next, the aforementioned model is designed. The BERT-Base (Uncased) layer has an input size of 400 tokens (clipping the input at 400 tokens and dropping the rest). The outputs of this layer are embedding vectors with a hidden size of 768 corresponding to the 400 input tokens. The second layer is an attention layer (attention matrix size = 4096) with a Relu activation function. Our last layers are two fully connected layers (4096 nodes) with dropout of 0.1. There are two neurons in the final layer, the objective of which is to detect whether the sentence is an instance of hate speech or not. The model is trained for 10 epochs with the Adam

<sup>1</sup><https://github.com/alirezasalemi7/SemEval2021-Toxic-Spans-Detection>

optimizer and a learning rate of 0.001. We freeze the weights of the BERT layer during this training process as we find through experimentation that fine-tuning BERT in this stage results in lower performance of our model in the toxic span detection task.

Once the model has been trained, we input our sentence and if our sentence-level detector predicts the sentence to be non-hateful we move on and produce a blank output as our toxic span. If, however, the model detects the sentence to be hateful, we extract the attention values and calculate the attention score of each word. If a word is made up of multiple subwords, we average the values of all subwords. After the attention scores have been calculated we use rule-based and machine learning models to label spans as toxic. These models are explained in Table 2. We begin by rule based models, selecting a percentage of spans with attention scores above a certain threshold (shown in Figure 2). Additionally, we test different machine learning models with various sets of features. Our results are shown in Section 5.3.

## 4.2 Named Entity-based Methods

Our second intuition is to look at this problem as one similar to NER. As such, our toxic span label can be looked at as another NER label. We considered toxic, non-toxic and padding as labels and applied CRF to this NER task. The padding label was added to reduce the model bias toward the non-toxic class.

Our model is depicted in Figure 1. We train the model for 2 epochs with a learning rate of  $3 \times 10^{-5}$ . In contrast to the previous method, the embedding layer is fine-tuned during our training process. Our tests on these models are shown in Section 5.4.

## 4.3 Ensemble Models

Finally, we test two methods of combining the outputs of various models in order to achieve a better performance on the task. As previously mentioned, the expected outputs of the task are numerical indexes of the parts of the string which are believed to be toxic. Consequently, the first method of mixing could be voting, where if the majority of the models vote for one index, the index is included in the final selection. The second method is based on calculating the intersection of outputted indexes of all three models. In other words, only adding an index if it is detected by all three models. The results are shown in Section 5.6.

# 5 Results

In this section we will report the results of the models introduced in the previous section (4) on the toxic span detection task. Per the competition evaluation instructions, for all models the F1 score is reported.

## 5.1 Random Baseline

To help us better understand the complexity of the task at hand, we start with a random baseline. In this method, we first split each sentence into words (using NLTK’s functions) and then randomly label each word as toxic or not. We observe that this baseline F1-score for the task is 0.17.

## 5.2 Keyword-based

The second simple method we test is a word-matching one. Our intuition is that toxic spans will likely include hateful or negative words. Thus we begin with a list of hate words and label any word found on the list as toxic and label the rest as nontoxic. This method results in an F1-score of 0.332 which is almost twice that of the random baseline, showing that while not all hate words are toxic and not all toxic spans are hate words, there is still a considerable amount of overlap. We further test if most words in toxic spans will have a negative sentiment value. Thus we repeat the same method, this time labeling anything with a negative sentiment as toxic. To detect the sentiment score of each word we use *TextBlob* (Loria, 2018). We see that this method achieves an F1 of 0.378, outperforming the aforementioned technique. Finally we mix the two methods (labeling both hate words and words with negative sentiment as toxic), and achieve an F1-score of 0.418.

## 5.3 Attention-based

As mentioned in Section 4, the intuition behind the attention-based model is that the model which learns to detect hate speech, would learn to pay more attention to the hateful spans in the text. Consequently, we test this idea in Table 2. We can see that the rule-based attention selection method outperforms other span selection techniques. To select the best set of rules for the model, we test both the percentage of top-words (with respect to attention) which we consider for selection, and the threshold we place on the minimum value of attention which is considered. As shown in Figure 2, we can see that the top 75% of attention scores with a thresh-

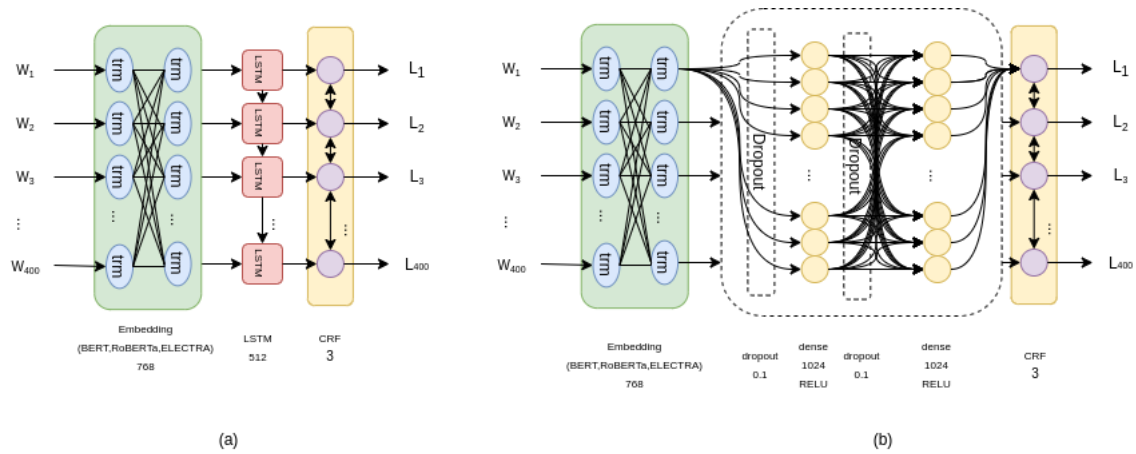


Figure 1: The architecture of the named entity-based models. (a) displays a version of the model in which the two dense layers in (b) have been replaced by an LSTM layer. The results of both versions are shown in Table 3.

old of  $10^{-4}$  is the best set of hyper-parameters for the task.

Upon analysis of the results of the attention-based model, we find that the model performs well on the detection of single word spans (detecting 78% of single-word spans in the evaluation dataset) but does not detect multi-word spans well (only detecting 16% of such spans completely). This is because the distribution of attention scores are observed to be such that there is a large focus on one word and other words receive little attention values.

We further set up another experiment where we assumed that the true sentence-level labels were given. The model then predicted the toxic spans given these true labels achieving an F1 of 0.808. This shows that if the sentence-level classifier performed better, our model would have been able to get higher performance. Thus, more focus should be placed on obtaining higher accuracy in the sentence-level classification task.

#### 5.4 Named Entity-based

Table ??, displays the results of our named entity based models. We can see that LSTM layers do not improve performance, and among various embeddings, RoBERTa outperforms the others in our 5-fold cross validation testings. However, BERT achieves better results in the competition’s evaluation phase.

#### 5.5 Google’s T5

Another model we test is Google’s T5 (Raffel et al., 2019). To test the T5 model, we use hugging-face’s

T5-base model<sup>2</sup> and frame our problem as one where the context is the Tweet text and the answer is the text of the toxic spans to be detected. Our model achieves an F1 of 0.635 in the evaluation phase of the competition.

#### 5.6 Ensemble Models

As described in section 4.3, we tested intersecting and using a voting scheme for the model outputs. More precisely, we perform these methods on the outputs of the following named entity based models:

- (1) BERT + CRF
- (2) Electra + CRF
- (3) RoBERTa + CRF

We find that the competition evaluation F1 reaches 0.681 when we use voting of indexes, and 0.684 when the indexes are intersected. As can be seen both methods outperform all individual models.

### 6 Conclusion

In this study we presented and compared various methods for toxic span detection. We examined the problem from various points of views reporting our results using each model. Our best system, an ensemble model, achieved an F1 of 0.684 in the SemEval-2021 Task 5 evaluation phase. Among the named-entity-based models, BERT+CRF performs best achieving an F1 of 0.67. Our attention-based model achieved an F1 of 0.609 in the competition’s

<sup>2</sup>We were not able to test a larger version of the model due to system constraints

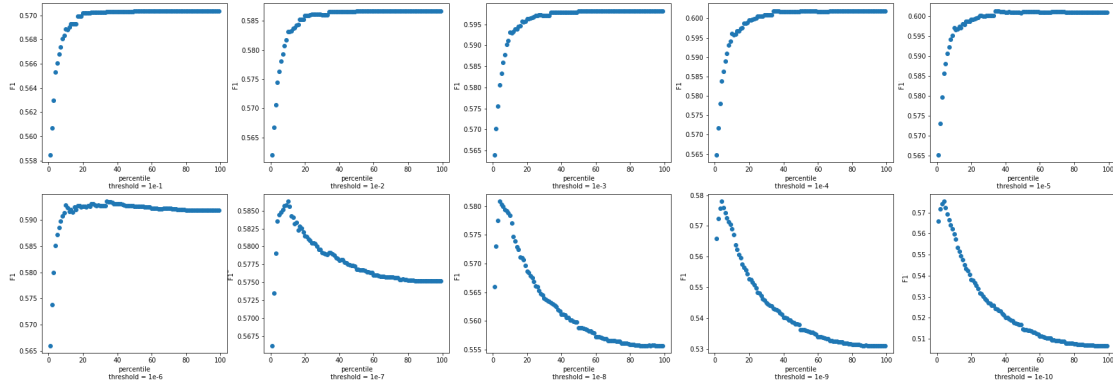


Figure 2: The effects of placing various thresholds on the minimum value of attention scores allowed to be selected and the percentage of top scores that have been selected on the F1-score of the toxic span detection task. Each plot displays one threshold value, and the x-axis in each plot is the percentile of scores we select and the y-axis is the F1 value achieved by this combination of threshold and percentile.

Model	Hate Speech Detection				Toxic Span Detection	
	Accuracy	Precision	Recall	F1	F1	F1 (Competition Evaluation)
Span Selection Rules						
R1 <sup>a</sup>	0.85	0.85	0.85	0.85	0.601	0.609
R2 <sup>b</sup>					0.601	-
R3 <sup>c</sup>					0.496	-
Decision Tree <sup>d</sup>					0.360	-
Neural Network <sup>e</sup>					0.354	-

- <sup>a</sup> **R1**: selecting words with top 75% of attention scores with threshold  $10^{-4}$  and then removing stop-words
- <sup>b</sup> **R2**: R1 + removing positive sentiment words among the top 75%
- <sup>c</sup> **R3**: R2 + adding all hate words (using the hate word list) in the sentence regardless of attention scores
- <sup>d</sup> **Decision Tree**: the input features of the model are: 1-attention score of word, 2-part of speech of the word, 3-sentiment of word 4-whether the word is a hate word or not (0/1)
- <sup>e</sup> **Neural Network**: the features inputted to the model are 1-attention score of word, 2-part of speech of the word, 3-sentiment of word, 4-whether the word is a hate word or not (0/1) - categorical features (e.g. POS) are modeled as learnable embeddings.

Table 2: Results of the attention-based models, the model structure is *BERT + Attention + Dense* and we have tested out different span selection rules

Embedding	Layers	F1 (train)	F1 (test)	F1 (Competition Evaluation)
BERT	CRF	0.702	0.648	0.67
RoBERTa	CRF	0.682	0.652	0.66
Electra	CRF	0.687	0.646	0.65
BERT	LSTM + CRF	0.668	0.62	-
RoBERTa	LSTM + CRF	0.669	0.647	-
Electra	LSTM + CRF	0.678	0.641	-

Table 3: Results of the named-entity based models evaluated using 5-fold cross-validation

evaluation phase. Future work could focus on the improvement of the sentence-level detection in our attention scheme, as we showed improvement in that regard would improve this task’s performance.

## References

Aymé Arango, Jorge Pérez, and Barbara Poblete. 2019. Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–54.

Valerio Basile, Cristina Bosco, Elisabetta Fersini,

- Nozza Debora, Viviana Patti, Francisco Manuel Rangel Pardo, Paolo Rosso, Manuela Sanguinetti, et al. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *13th International Workshop on Semantic Evaluation*, pages 54–63. Association for Computational Linguistics.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019a. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 491–500.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019b. [Nuanced metrics for measuring unintended bias with real data for text classification](#). *CoRR*, abs/1903.04561.
- Cristina Bosco, Dell’Orletta Felice, Fabio Poletto, Manuela Sanguinetti, and Tesconi Maurizio. 2018. Overview of the evalita 2018 hate speech detection task. In *EVALITA 2018-Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian*, volume 2263, pages 1–9. CEUR.
- Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. *arXiv preprint arXiv:2010.03260*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Ashwin Geet D’sa, Irina Illina, and Dominique Fohr. 2019. Towards non-toxic landscapes: Automatic toxic comment detection using dnn. *arXiv preprint arXiv:1911.08395*.
- Mai ElSherief, Vivek Kulkarni, Dana Nguyen, William Yang Wang, and Elizabeth Belding. 2018. Hate lingo: A target-based linguistic analysis of hate speech in social media. *arXiv preprint arXiv:1804.04257*.
- Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30.
- Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N Asokan. 2018. All you need is” love” evading hate speech detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, pages 2–12.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Jigsaw-Conversation-AI. *Detect toxicity across a diverse range of conversations*. <https://cutt.ly/XhOYKPR>.
- Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.
- Steven Loria. 2018. textblob documentation. *Release 0.15*, 2.
- Binny Mathew, Anurag Illendula, Punyajoy Saha, Soumya Sarkar, Pawan Goyal, and Animesh Mukherjee. 2020. Hate begets hate: A temporal study of hate speech. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW2):1–24.
- Mainack Mondal, Leandro Araújo Silva, and Fabrício Benevenuto. 2017. A measurement study of hate speech in social media. In *Proceedings of the 28th acm conference on hypertext and social media*, pages 85–94.
- Usman Naseem, Imran Razzak, and Peter W Eklund. 2020. A survey of pre-processing techniques to improve short-text quality: a case study on hate speech detection on twitter. *Multimedia Tools and Applications*, pages 1–28.
- nicapotato. *Bad Bad Words*. <https://www.kaggle.com/nicapotato/bad-bad-words>.
- JT Nockleby. 2000. ‘hate speech in encyclopedia of the american constitution.
- Sean Papay, Roman Klinger, and Sebastian Padó. 2020. Dissecting span identification tasks with performance prediction. *arXiv preprint arXiv:2010.02587*.
- John Pavlopoulos, Ion Androutsopoulos, Jeffrey Sorensen, and Léo Laugier. *Toxic Spans Detection*. <https://sites.google.com/view/toxicspans>.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5: Toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Alain Pinsonneault and Nelson Heppel. 1997. Anonymity in group support systems research: A new conceptualization, measure, and contingency framework. *Journal of Management Information Systems*, 14(3):89–108.
- Fabio Poletto, Valerio Basile, Manuela Sanguinetti, Cristina Bosco, and Viviana Patti. 2020. Resources and benchmark corpora for hate speech detection: a systematic review. *Language Resources and Evaluation*, pages 1–47.

- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International workshop on natural language processing for social media*, pages 1–10.
- Leandro Silva, Mainack Mondal, Denzil Correa, Fabrício Benevenuto, and Ingmar Weber. 2016. Analyzing the targets of hate in online social media. *arXiv preprint arXiv:1603.07709*.
- Matthew Tang, Priyanka Gandhi, Md Ahsanul Kabir, Christopher Zou, Jordyn Blakey, and Xiao Luo. 2019. Progress notes classification and keyword extraction using attention-based deep learning models with bert. *arXiv preprint arXiv:1910.05786*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008.
- Xuan-Son Vu, Thanh Vu, Mai-Vu Tran, Thanh Le-Cong, and Huyen Nguyen. 2020. Hsd shared task in vlsp campaign 2019: Hate speech detection for social good. *arXiv preprint arXiv:2007.06493*.
- Lu Xu, Lidong Bing, Wei Lu, and Fei Huang. 2020. Aspect sentiment classification with aspect-specific opinion spans. *arXiv preprint arXiv:2010.02696*.

# macech at SemEval-2021 Task 5: Toxic Spans Detection

**Maggie Cech**

SAS Institute

margaret.cech98@gmail.com

## Abstract

Toxic language is often present in online forums, especially when politics and other polarizing topics arise, and can lead to people becoming discouraged from joining or continuing conversations. In this paper, I use data consisting of comments with the indices of toxic text labelled to train an RNN to determine which parts of the comments make them toxic, which could aid online moderators. I compare results using both the original dataset and an augmented set, as well as GRU versus LSTM RNN models.

## 1 Introduction

In this digital era we live in, almost everyone is communicating online. As of January 2021, Facebook, YouTube, and WhatsApp each have over 2 billion users, which means many differing viewpoints and perspectives being shared (Statista, 2021). With such a huge exchange of ideas, there is bound to be some toxicity within the comments. Aside from discouraging users to continue with or join conversations, toxic comments can also taint users' perceptions on news sites (Tenenboim et al., 2019). Thus it is important to moderate online conversations without fully censoring users.

While forums typically rely on human moderators, with such vast amounts of data coming in, it can be difficult for humans to keep up (Nobata et al., 2016). Advances in deep learning and machine learning is making text processing a viable option to replace, or at least assist, human moderators clean up comment sections (Consultants, 2019). Some methods rely on simply classifying whether a comment is toxic or not, but identifying what parts of the text are actually toxic can assist moderators and provide insight into what makes language toxic. The SemEval task 5 aims to evaluate systems that detect toxic spans within text using

datasets where spans within the comments are labelled as toxic, differing from previously released datasets where whole comments were labelled as toxic or non-toxic (Pavlopoulos et al., 2021).

This is inherently a natural language processing task, similar to text classification and sentiment analysis. This study focuses on training a recurrent neural network to determine the indices of a given string that represent the toxic portions of a comment. Recurrent neural networks are classically used for natural language and sequence labelling task, and one could view this task as a form of sequence labelling. The goal of sequence labelling is, given a sequence as input, assign a sequence of labels. Because recurrent neural networks (RNNs) are flexible in their use of context information and can recognize sequential patterns, they are an attractive and commonly used choice in sequence labelling (Graves, 2012). This paper approaches the task at hand with a sequence labelling methodology, applying an RNN and comparing the use of gated recurrent unit (GRU) and long-short term memory unit (LSTM) layers in the RNN.

## 2 Related Work

Aggression in text is complex, often clouded by sarcasm or including repeat words that cause a model to incorrectly identify words as toxic. A study by Vaidya, Mai, and Ning found that comments including identities, such as LGBTQ+, Black, Muslim, and/or Jewish identities, often resulted in false positives for toxic comments, so this was a bias we wanted to be aware of in our study (Vaidya et al., 2020).

Detecting toxic spans is not as common of a task as toxic comment detection or sentiment analysis. Many studies surrounding toxic comments have been completed largely in part due to the availability of a large corpora of data released by the

Wikimedia Foundation, as well as several Kaggle competitions hosted by Google Jigsaw. Other studies generally take a text classification approach similar to sentiment analysis, which as previously mentioned, is not exactly the task at hand (Nobata et al., 2017).

One of the comparisons made in this research is between GRU and LSTM recurrent models. Generally, LSTM units have issues with vanishing gradients when text sequences are too long, so GRUs are used instead. GRU controls the flow of information like the LSTM unit, but without the use of a memory unit (Chung et al., 2014). Previous research has shown GRU outperforms LSTM for all depths in speech recognition. The same study determined that bi-directional RNN models consistently outperform uni-directional models and found that models with 5 or more recurrent layers did not improve the results (Khandelwal et al., 2016). As a result, in our research, we compare GRU and LSTM models to find if GRU will consistently outperform LSTM in a sequence labelling task, as opposed to speech recognition, and use bi-directional RNN models with fewer recurrent layers, as they were not shown to have any benefit. This paper will also compare the results of augmented and non-augmented datasets to determine if the use of synonyms will improve the performance after training.

### 3 Methodology

#### 3.1 Pre-Processing

Quite a bit of pre-processing was required to prepare this data for training. First, we label the comments using the given indices representing the toxic spans within the comments. A word labelled with a "/1" is toxic and with "/0" is not toxic. The comments are then run through a function that goes through the following text preprocessing steps:

1. Text to lowercase
2. Remove URLs
3. Remove numbers
4. Remove extra whitespace
5. Expand contractions ("it's" to "it is", "they're" to "they are", etc.)
6. Remove punctuation

7. Tokenize the strings - here one token is one word in the comment
8. Lemmatize the tokens
9. Remove stop words (list of stop words from nltk.corpus package)

After this text processing is completed, we use SAS DLPy<sup>1</sup> and SAS SWAT Python<sup>2</sup> packages to continue with the processing. From the SAS SWAT package, we are able to connect to a server where we can upload the cleaned text data into a CASTable (similar to a Pandas DataFrame). Then the text data needs to be converted to embeddings. To do this, we use GLOVE 100-dimension trained word vectors and apply the vectors to the CASTable containing the comments. Because text data from human sources is so varied and can often include unknown words or misspellings, we remove any comments that could not be converted to numeric embeddings and place them in a separate table. Instead of using these unknown embeddings in training and scoring using a neural network, we will make simple predictions based on if any common toxic words are present in the comments. This is not ideal but only about 2-5% of data ends up in this separate table and saves us the trouble of dealing with finding all of the words that could not be converted to embeddings.

Ideally we would not be removing observations from the dataset and further experimentation with different embedding sets could lead to better outcomes. This does affect the final results slightly as there is both less data being used in training and worse prediction accuracy from the comments not used in the main prediction set contributing to the final score, but because the percentage of data actually removed is so low, we decided it would not make a big enough difference to focus on. The comments removed from the main set are predicted using common toxic words gathered from the training set. Although misspellings are often a cause for the embeddings failing on a comment, the misspelled word may not have been part of the toxic span, so the comment could still be predicted using common toxic words and have a fairly acceptable accuracy.

After the comments are converted to embeddings, we find the maximum column count of a

<sup>1</sup><https://sassoftware.github.io/python-dlpy/>

<sup>2</sup><https://sassoftware.github.io/python-swat/>



single embedding and pad all other embeddings to this maximum value. We then create output columns - one column for each 100-D word vector - and fill the columns with the toxicity labels from earlier. The final table used for training and scoring contains columns for the original comment, the labelled comment, the cleaned comment, 100 dimension embeddings for each word in the comment and the rest of the columns zero-padded to the longest value, and toxicity labels for each word in the comment, also zero-padded to the longest value.

We can then build an RNN using SAS' deepLearn actionset<sup>3</sup>, train the model using the dlTrain action, and score using the dlScore action. When training, the embeddings columns are used as input columns to the model, where each token is 100 columns for each 100 dimension embedding, and the target columns are the label columns that contain either a 1 for a toxic word or a 0 for a non-toxic word, where each token is one column.

### 3.2 Augmented Data

The goal of using augmented data was to increase the amount of data being used in training the models. To do this, we found every comment with only one toxic word and created up to five new comments with the toxic word replaced with a different synonym. We were able to do this by using wordnet from the nltk Python package that finds a list of synonyms for a given word. Figure 1 shows

	original comment
0	Another violent and aggressive immigrant killi...
1	I am 56 years old, I am not your fucking junio...
2	<b>Damn, a whole family. Sad indeed.</b>
3	What a knucklehead. How can anyone not know th...
4	"who do you think should do the killing?"\n\nA...

Figure 1: Original toxic comments

the original comments, with comment 2 being the example that is augmented, and figure 2 shows the new comments created using synonyms of the

	original comment
6	<b>damn,</b> a whole family. Sad indeed.
7	<b>dam,</b> a whole family. Sad indeed.
8	<b>hoot,</b> a whole family. Sad indeed.
9	<b>red_cent,</b> a whole family. Sad indeed.
10	<b>shit,</b> a whole family. Sad indeed.

Figure 2: Augmented toxic comments

word "damn". By augmenting only comments that contain a single toxic comment, we are able to increase the size of the data set from 7,939 comments to 21,822 comments - almost three times as many comments to be used in training. The idea here was to increase the size of the training data set to improve performance.

### 3.3 GRU vs LSTM

As mentioned in the introduction, we trained a bi-directional RNN model. After comparing performance, we found that a model with more than one bi-directional RNN layer was not improving the accuracy of the predictions, so we used a smaller model with only one bi-directional layer. The input layer connects to a fully connected layer, which goes into the bi-directional layer, into another fully connected layer, and finally to the output layer. Figure 3 shows the model architecture. We trained our data on two different models, one with a GRU cell used in the bi-directional layer and one that uses an LSTM cell in the bi-directional layer.

## 4 Experimental Results

In this section, we discuss the results from the two experiments - GRU vs LSTM models and augmented vs original data used in training. We use F1 scores to compare the two different models and types of data sets. The F1 scores, discussed later on, are a combination of both the predictions from the trained models and the comments that are predicted using common toxic words (which will henceforth be referenced as "guessed comments" for lack of better terms). The guessed comments do have a slight effect on the final results which will be described in this section

<sup>3</sup>[https://go.documentation.sas.com/doc/en/pgmsascdc/9.4\\_3/casdlpg/cas-deeplearn-TblOfActions.htm](https://go.documentation.sas.com/doc/en/pgmsascdc/9.4_3/casdlpg/cas-deeplearn-TblOfActions.htm)

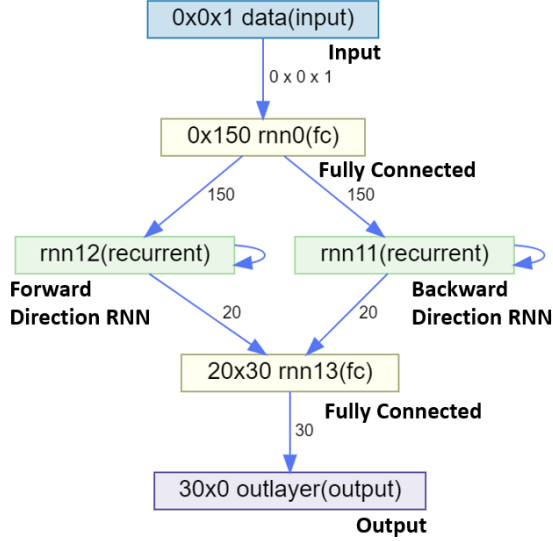


Figure 3: RNN model architecture showing layer dimensions

#### 4.1 Evaluation Metric

Each model’s performance was evaluated using an F1 score, as described in (Martino et al., 2019). If the system is represented by  $A_i$ , the return set from the system is  $S_{A_i}^t$ ,  $t$  represents a post or comment, and  $G$  represents the ground truth annotations for post  $t$ , then F1 score of a system is defined as:

$$F_1^t(A_i, G) = \frac{2 \cdot P^t(A_i, G) \cdot R^t(A_i, G)}{P^t(A_i, G) + R^t(A_i, G)} \quad (1)$$

$$P^t(A_i, G) = \frac{|S_{A_i}^t \cap S_G^t|}{S_{A_i}^t} \quad (2)$$

$$R^t(A_i, G) = \frac{|S_{A_i}^t \cup S_G^t|}{S_{A_i}^t} \quad (3)$$

If  $S_G^t = 0$ , an instance where there are no toxic spans present, then  $F_1^t(A_i, G) = 1$  if no toxic spans are predicted, and  $F_1^t(A_i, G) = 0$  otherwise. To obtain the final F1 score for a particular system, the F1 scores are averaged over all posts  $t$ .

#### 4.2 Augmented Data

Table 1 shows comparisons between the use of augmented and non-augmented data when training both the GRU and LSTM. We can see in the various hyperparameter settings, the augmented data actually performs worse than the original data. This is interesting considering generally the more data points used, the better the model. It is possible that

the method in which we augmented the data, only changing one word within a comment, did not help the model learn any of the connections between the text but rather memorize more toxic words, many of which are often repeated.

Table 1 also shows that the learning rate does not affect the augmented data quite as drastically as it does the non-augmented data, with the non-augmented data showing much lower F1 scores when a lower learning rate is used, such as 0.001. It may be interesting to further explore if larger datasets are less affected by changes to the learning rate. The augmented dataset is almost three times larger than the original dataset, so it is also much more time consuming to train. With much longer training times and worse performance, the original dataset proves to be the more effective option. Table 3 shows that across both dev and test datasets, results from augmented dataset training were worse than their non-augmented counterparts.

We also compared results from only comments with a single toxic word present to find if these performed better for the augmented data, since only comments with a single toxic word were augmented. Again, the original dataset outperformed the augmented data. The differences between the single toxic word scores and scores when all comments are factored in are also very similar between augmented and non-augmented data results, showing the augmented data did not make much of a difference in predicting single toxic word comments. Table 2 shows these results.

#### 4.3 GRU vs LSTM

After tuning the hyperparameters for both models, we found that the GRU model performs slightly better than LSTM across most hyperparameter settings. We were able to get the highest F1 score using GRU as well. Even between the scores using augmented data, GRU still performs slightly better than LSTM. Table 1 shows these results.

#### 4.4 Results and analysis

Table 3 shows F1 scores using the best hyperparameter settings across the four methods - GRU, GRU trained with augmented data, LSTM, and LSTM trained with augmented data. The F1 test column shows scores for the evaluation data given, which was to be used for submitting a team’s results. Our team submitted results for each of these methods in hopes of securing a better spot on the leaderboard, but ultimately ended up with a very low score, the

Max Epochs	Learn Rate	Mini-batch Size	GRU	GRU Augmented	LSTM	LSTM Augmented
10	0.1	5	0.515	0.504	0.514	0.501
10	0.05	5	0.514	0.503	0.499	0.495
10	0.001	5	0.287	0.406	0.324	0.393
10	0.1	10	0.473	0.459	0.463	0.469
50	0.1	5	0.518	0.478	0.512	0.460
20	0.1	5	0.519	0.494	0.504	0.464
20	0.05	5	0.369	0.433	0.396	0.513
20	0.001	5	0.521	0.500	0.505	0.501

Table 1: F1 scores during hyperparameter tuning for GRU and LSTM models with both original and augmented data

Max Epochs	Learn Rate	Mini-Batch Size	Single Toxic Original	All Toxic Original	Single Toxic Augmented	All Toxic Augmented
10	0.1	5	0.622	0.515	0.594	0.504
10	0.05	5	0.623	0.514	0.602	0.503
10	0.001	5	0.351	0.287	0.489	0.406
10	0.1	10	0.542	0.473	0.530	0.459
50	0.1	5	0.610	0.518	0.561	0.478
20	0.1	5	0.623	0.600	0.583	0.494
20	0.05	5	0.451	0.369	0.528	0.433
20	0.001	5	0.616	0.521	0.601	0.500

Table 2: F1 scores during hyperparameter tuning comparing performance for only single toxic comments and all toxic comments using the GRU model

Method	F1 Dev	F1 Test
GRU	0.519	0.602
GRU Augmented	0.504	0.551
LSTM	0.514	0.600
LSTM Augmented	0.501	0.550

Table 3: Best F1 scores for dev and test sets

reasoning for which is described in the note below. We were able to later calculate the F1 scores for the evaluation data when the entire dataset was released, after the competition ended, and those are the scores shown in Table 3.

It is interesting to note that the test data outperforms the dev data for every method used. This may be due to some kind of overfitting occurring. This is peculiar but is also likely due to the evaluation data containing fewer guessed comments, or comments predicted using common toxic words instead of predicted by the model. Because both the predicted comments and the guessed comments are used in calculating the F1 score, the portion

of the entire dataset that is guessed would have an impact on the final score. The dev set had 2.5% of guessed comments and the test set had slightly fewer, with 2% of the entire dataset being comprised of guessed comments, which would explain the better results shown.

One can also see in Table 3 that the GRU model outperforms the LSTM model, even if just slightly, for both the original and augmented datasets and across both dev and test data. Other research has noted that GRU cells may be better for specificity, or finding true negatives, and focusing on less prevalent content, whereas LSTM cells are better for detecting true positives and focusing on highly prevalent content. (Gruber and Jockisch, 2020). Looking into the dev dataset, only 6% of the observations contain no toxic spans, but this model is not predicting whether an entire comment contains any toxic spans, it is predicting if each word is toxic. Out of all of the text, 93% of the words are non-toxic words, or in this case, words to be labelled negative for toxicity. A model that can better predict true negative outcomes would have

the best performance with this data.

#### 4.5 Note on Official Results Discrepancy

The official results in task 5 for the SemEval conference show this team to have an F1 score of 0.070. This score is much lower than the values we are presenting here because after adding the ground-truth values for the evaluation data it was found that the best F1 score we achieved was 0.602, as shown in Table 3, and the results were submitted to the competition out of order or formatted incorrectly, producing a very low F1 score.

#### 5 Conclusion

We experimented with several different RNN models and ultimately utilized the results from bi-directional GRU and LSTM models. It was found that the GRU model slightly outperforms the LSTM model for this test case. As was found by Gruber and Jockisch, GRU cells can be better for detecting true negatives and since 93% of the words in the training dataset we used were non-toxic words, it follows that GRU cells may outperform LSTM cells in cases where there are more negative instances than positive in the datasets. It would be interesting to explore if using both a GRU cell and an LSTM cell in a model would further increase performance, with GRU focusing on less prevalent content and true negatives, and LSTM focusing on high prevalent content and true positives.

In the exploration of the use of augmented data, we found that not only did the models trained on augmented data perform worse than those trained on the original dataset, but they were also much slower to train. We compared if the F1 scores for observations with a single toxic word were higher for the models trained with augmented data, since only comments with a single toxic word present were augmented, but still the original datasets outperformed the augmented datasets. Because the augmented data does not improve results, we can continue to use the original dataset to cut down on computation time. We also found that the F1 scores resulting from the augmented data training did not react as greatly to changes in the learning rates as the original datasets did, which may be interesting to explore further.

#### References

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. [Empirical evaluation of](#)

[gated recurrent neural networks on sequence modeling](#). arXiv:1412.3555v1.

Cambridge Consultants. 2019. [Use of ai in online content moderation](#).

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Publishing.

Nicole Gruber and Alfred Jockisch. 2020. [Are gru cells more specific and lstm cells more sensitive in motive classification of text?](#) *Frontiers in Artificial Intelligence*, 3(40).

Shubham Khandelwal, Benjamin Lecouteux, and Laurent Besacier. 2016. [Comparing gru and lstm for automatic speech recognition](#).

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news article. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China. Association for Computational Linguistics.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user contents. In *Proceedings of the 25th International Conference on World Wide Web*, pages 145–153.

Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2017. [Personal attacks seen at scale](#). In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

John Pavlopoulos, Leo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. Semeval-2021 task 5:toxic spans detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.

Statista. 2021. [Global social networks ranked by number of users 2021](#).

Ori Tenenboim, Gina M. Masullo, and Shuning Lu. 2019. [Attacks in the comment sections: what it means for news sites](#). Technical report, The University of Texas at Austin Center for Media Engagement.

Ameya Vaidya, Feng Mai, and Yue Ning. 2020. [Empirical analysis of multi-task learning for reducing identity bias in toxic comment detection](#). arXiv:1909.09758.

# LZ1904 at SemEval-2021 Task 5: Bi-LSTM-CRF for Toxic Span Detection using Pretrained Word Embedding

**Liang Zou**

Department of Mathematics  
New York University  
lz1904@nyu.edu

**Wen Li**

Department of Linguistics  
Indiana University  
wl19@indiana.edu

## Abstract

Recurrent Neural Networks (RNN) have been widely used in various Natural Language Processing (NLP) tasks such as text classification, sequence tagging, and machine translation. Long Short Term Memory (LSTM), a special unit of RNN, has the advantage of memorizing past and even future information in a sentence (especially for bidirectional LSTM). In the shared task of detecting toxic spans in texts, we first apply pretrained word embedding (GloVe) to generate the word vectors after tokenization. Then we construct Bidirectional Long Short Term Memory-Conditional Random Field (Bi-LSTM-CRF) model by Baidu research to predict whether each word in the sentence is toxic or not. We tune hyperparameters of dropout rate, number of LSTM units, embedding size with 10 epochs and choose the epoch with best validation recall. Our model achieves an F1 score of 66.99% on test dataset.

## 1 Introduction

Detecting toxic words plays a critical role in social media to ensure healthy online discussions. In previous study, some tasks (Liu et al., 2019; Borkan et al., 2019a) only identify offensive language based on the whole sentence or post. Most of them do not detect specific spans of words that make the sentence or post offensive.

In SemEval-2021 Task 5: Toxic Spans Detection (Pavlopoulos et al., 2021), the data was collected from civil comments (Borkan et al., 2019b). Each post is in string format, and a word is marked as toxic span in the form of its characters' offsets in the string. The goal of the task is to classify whether each word in a sentence is toxic or not. If so, the indices of characters in the word should be returned. The task is evaluated by F1 score based on the character offsets among all posts.

The challenges of this task include:

- The small dataset makes it very difficult to train complicated models like deep neural networks, since it may cause overfitting.
- We need to predict which word or phrase is toxic given a text (many-to-many) rather than whether the entire sentence is offensive or not (many-to-one). This creates restrictions on feature engineering and modeling:
  - Feature Engineering: We cannot delete or add words in the sentence.
  - Modeling: Models need to be specific on each word instead of sentiment classification on whole sentence.
- Most of the words and phrases in sentences are not toxic. This indicates our dataset is imbalanced.

The models we explore in this task include word-based Conditional Random Field (CRF), word-based Bidirectional Long Short Term Memory (Bi-LSTM) with and without pretrained word embedding, Bidirectional LSTM-CRF with pretrained word embedding. We choose Bi-LSTM-CRF as final submission, since it performs the best during our experiments.

The structure of this paper is organized as follows:

- In section 2, we review related work of applications of different models in Sequence Tagging, Name Entity Recognition, and Sentiment Analysis.
- In section 3, we present the summary statistics of data and how we build models with performance evaluation.
- In section 4, we discuss our model results on validation dataset with key findings.

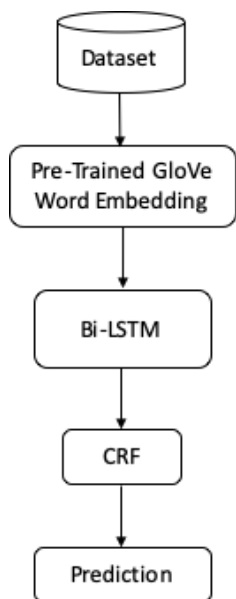


Figure 1: System Flowchart

- In section 5 and 6, we present our conclusion based on experiment results and future work.

## 2 Related Work

Toxic Span Detection is a type of sequence labeling tasks and is similar to name entity recognition tasks with only two categories. (Xing et al., 2010) surveyed various sequence labeling tasks in terms of methodologies and applications. They also reviewed a few extensions of sequence classification including early classification and semi-supervised learning on sequences. (Nguyen and Guo, 2007) compared different learning algorithms such as Conditional Random Fields (CRF), Support Vector Machine (SVM), and Perceptron for sequence labeling tasks. (Akbik et al., 2018) proposed a new type of embedding called contextual string embedding for sequence labeling tasks. A Comparison between multiple word embedding methods were conducted by (Lauren et al., 2018) for sentiment classification and sequence labeling tasks.

For name entity recognition (NER), (Mansouri et al., 2008) presented a machine learning based approach called Fuzzy Support Vector Machine. Recent advanced deep learning models were summarized by (Yadav and Bethard, 2019) in various shared tasks.

Currently, models that are widely used in various NLP tasks include CRF, LSTM, RNN, and transformers. Also, word embedding such as Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al.,

Statistics	Train	Trial
count	7939	690
mean	43	41
std	41	40
min	1	1
25%	16	15
50%	29	28
75%	54	53
max	240	217

Table 1: Distribution of word count

Type	Train	Trial
Toxic	24135	1814
Non toxic	1881225	163786

Table 2: Count of toxic and non toxic words in data

2014), and ELMo (Peters et al., 2018) are preferred before model training.

## 3 Data and Methodology

### 3.1 Data Description

Toxic Spans Detection Dataset includes trial and training data. The training data contains 7939 records, and trial data contains 690 records. The sentences and indices of characters in toxic span are provided separately. To obtain if a word is toxic or not after tokenization, we split the text by space and punctuation and map the indices of toxicity to corresponding words. As a result, the word sequences in text will be marked as toxic (1) or non-toxic (0).

The distribution of the length of text (in word count) is summarized in Table 1. It shows that training and trial data follow a similar distribution in percentile, mean, and standard deviation (std).

The count of toxic and non-toxic words in texts are concluded in Table 2. It shows the dataset is highly imbalanced that most of words are non toxic.

### 3.2 Methodology

**CRF** Conditional Random Field (CRF) was developed in 2001 (Lafferty et al., 2001) for sequence prediction. Given the observable sequence  $X$  and labeling sequence  $Y$ , the objective of CRF is to construct model for conditional probability  $P(Y | X)$ . An advantage of using CRF compared with other sequential models such as Hidden Markov Model (Rabiner and Juang, 1986) is that it does not rely on the assumption of label independence.

Model	Precision	Recall	F1
CRF	0.686	0.341	0.4556
Bi-LSTM	0.6277	0.3664	0.4627
Bi-LSTM (glove-twitter-50)	0.8571	0.4884	0.6222
Bi-LSTM (glove-twitter-100)	0.8113	0.5	0.6187
Bi-LSTM-CRF (glove-twitter-100)	0.8333	0.5233	0.6429

Table 3: Model evaluations on validation data

Before fitting the model, for each word we create binary features to check whether the word is uppercase, lowercase, titlecase, and digit. We also append the same features of previous and next words. Next, we use “crfsuite” package to build CRF model. We choose “lbfgs” as optimization algorithm, and we set c1 and c2 equal to 0.1, max iteration equal to 100.

**Bi-LSTM** Long Short Term Memory (LSTM) is one of the most commonly used recurrent neural networks in many natural language processing tasks (Hochreiter and Schmidhuber, 1997). It consists of input gate, output gate, forget gate, and cell. Its gate structure enables the model to memorize long-term dependency and to prevent gradient vanishing issues.

In the experiments, we use Bidirectional LSTM (Bi-LSTM) in tensorflow.keras as second baseline. We configure number of LSTM units to be 200, embedding size equal to 50, and max sequence length to be 240, which is the max sentence length in training dataset. Thus, sentences with length of less than 240 will be padded. To reduce overfitting of neural networks, we set dropout rate as 0.2. Our final output layer uses sigmoid activation function with adam optimizer for gradient descent (Kingma and Ba, 2014).

We set number of epochs equal to 10 and record checkpoints. Since tensorflow package does not contain built-in F1 score, the final model parameters are loaded from the checkpoint with highest validation recall.

#### Bi-LSTM with pretrained word embedding

To further improve model performance, we adopt pretrained word embedding to generate word representation before training Bi-LSTM. The word embedding we use includes glove-twitter-50 and glove-twitter-100 in gensim (Řehůřek and Sojka, 2010). This means we need to modify our embedding size to 50 and 100 respectively. All other hyperparameters are consistent with Bi-LSTM above.

#### Bi-LSTM-CRF with pretrained word embedding

Our final model is Bidirectional LSTM-CRF created by Baidu Research (Huang et al., 2015). Compared with previous Bi-LSTM architecture, we add an extra output layer of CRF to make final predictions (as shown in Figure 1). Accordingly, we replace the loss function of binary cross entropy by CRF loss. We use glove-twitter-100 as pretrained embedding layer. All other hyperparameters of LSTM remain the same.

## 4 Experiment Results

We split proportion of training and validation dataset into 9:1. The evaluation results on validation set are summarized in Table 3. For each model discussed above, we list its precision, recall, and F1 score with default threshold. Since the dataset is highly imbalanced, we only focus on the evaluation metrics of toxic words. There are several key findings:

- Models with pretrained word embedding perform better than those without pretrained word embedding, since it produces higher precision and recall (thus higher F1 score).
- The performances of pretrained word embedding are close to each other regardless of embedding size. We do not want to further increase the embedding size, since it will increase the training time but not boost the performance significantly.
- As a final output layer, CRF can further improve recall for Bi-LSTM while keeping the precision in the same level. Therefore, it can increase F1 score.

Based on the model evaluation table, we choose Bi-LSTM-CRF with pretrained glove-twitter-100 embedding as our final model. The model achieves an F1 score of 0.6699 in final submission.

The confusion matrix for test data can be found in Table 5. We first flatten the sequence to a list of

Texts	Predictions	Error Type
Chris Birch is a mean, self-centered, contrary <u>ass</u> .		
... always <u>sucks</u> up to Big Oil.	[ass, sucks]	False Positive
I wish this <u>moron</u> would have been shot to death by the US soldier instead of the other way around.	[moron]	False Positive
Lord have Mercy on us, Trump is running <b>amok</b> .	[]	False Negative
... They're <b>vandals</b> , <b>thieves</b> , and bullies.	[]	False Negative

Table 4: Examples of False Positive / Negative

	Actual Pos	Actual Neg
Pred Pos	1747	1504
Pred Neg	736	73892

Table 5: Confusion matrix on test data

words before calculation. We define toxic words as positive (Pos) examples and non toxic words as negative (Neg) examples in the matrix. The table shows there are a lot of false positives (1504) in our model, this implies our model may be over-sensitive to the toxic words.

To deep dive into the model performance with specific examples, we collect a few sentences from test data in Table 4. In false positive examples marked as underline, the words “ass”, “sucks”, and “moron” are predicted as toxic words where there exists no toxicity in these sentences. In false negative examples marked as bold, the model fails to identify toxic words like “amok”, “vandals”, and “thieves”. The errors may come from the following reasons:

- Incorrect labels by ground-truth spans. These errors are unavoidable from the model due to human mistake.
- The pre-trained word embedding from GloVe does not reflect sentiment for those words. In other words, these words are not marked as positive or negative but neutral in word embedding.
- The position of words in sentence was not detected as toxicity by our Bi-LSTM-CRF model. For example, one word could be marked as toxic spans when it is in the beginning of the sentence but not the case when it is at the end. This will cause difficulty for model training to detect toxicity.

## 5 Conclusion

Detecting toxic words in texts is critical to furnish a healthy environment on social media. Sequence labeling task for finding specific offensive words is more difficult than sentiment classification on sentence level, since it requires models to locate the positions or indices of words in sentences. In addition, the task also places restrictions on feature engineering, because we cannot delete or add words in sentences.

Our experiment shows pretrained word embedding can improve model performance compared with randomized embedding weights. This verifies the concept of transfer learning where we can borrow the outputs from other resources and use them as inputs to achieve specific goals. Another finding is the benefit of model stacking where we add an extra layer of CRF after Bi-LSTM that further enhances predictability. In such case, when a single model does not work well in NLP tasks, combining different models with pretrained word embedding can be a good option to explore. However, there are still a lot of false positive examples in test set where the model predicts toxic words that in fact are not toxic.

## 6 Future Work

Further improvements can focus on feature engineering and model implementation. For feature engineering, we can conduct data augmentation for false negative examples: We first collect the words that are predicted as non-toxic but actually toxic, and reconstruct sentences using those toxic words as more training samples. This method can increase the weights of words that were originally omitted by the model, so that it may return better results. Similarly, we can also collect false positive examples and perform data augmentation to reduce false positive rates.

In addition to data augmentation, one can perform word-level text normalization to transform



words of different tenses to the same, even though each word cannot be deleted or added in a sentence.

From the model perspective, we may consider using more advanced classifiers with complicated structures. Due to resource limitations, we cannot design any large neural networks models such as deep neural networks (DNN) or transformers. Most of our experiments are done locally or via Google Colab. Training large neural networks will be very time-consuming and expensive when using tremendous amount of computing resources including multiple GPUs, TPUs.

If we have more time and available resources, we can experiment with more complex models such as BERT (Devlin et al., 2018), GPT (Radford et al., 2018), and so forth. In addition, we can deploy larger LSTM-related architecture including Bi-LSTM-CNN-CRF for sequence labeling (Ma and Hovy, 2016).

## References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th international conference on computational linguistics*, pages 1638–1649.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019a. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion proceedings of the 2019 world wide web conference*, pages 491–500.
- Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. 2019b. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data.
- Paula Lauren, Guangzhi Qu, Jucheng Yang, Paul Watta, Guang-Bin Huang, and Amaury Lendasse. 2018. Generating word embeddings from an extreme learning machine for sentiment analysis and sequence labeling tasks. *Cognitive Computation*, 10(4):625–638.
- Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 87–91.
- Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF. *arXiv preprint arXiv:1603.01354*.
- Alireza Mansouri, Lilly Suriani Affendey, and Ali Mamat. 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2):339–344.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Nam Nguyen and Yunsong Guo. 2007. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th international conference on Machine learning*, pages 681–688.
- John Pavlopoulos, Léo Laugier, Jeffrey Sorensen, and Ion Androutsopoulos. 2021. SemEval-2021 task 5: Toxic Spans Detection (to appear). In *Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Lawrence Rabiner and Biinghwang Juang. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.
- Zhengzheng Xing, Jian Pei, and Eamonn Keogh. 2010. A brief survey on sequence classification. *ACM Sigkdd Explorations Newsletter*, 12(1):40–48.
- Vikas Yadav and Steven Bethard. 2019. A survey on recent advances in named entity recognition from deep learning models. *arXiv preprint arXiv:1910.11470*.

# LIIR at SemEval-2021 task 6: Detection of Persuasion Techniques In Texts and Images using CLIP features

Erfan Ghadery , Damien Sileo , and Marie-Francine Moens

Department of Computer Science (CS)

KU Leuven

{erfan.ghadery, damien.sileo, sien.moens}@kuleuven.be

## Abstract

We describe our approach for SemEval-2021 task 6 on detection of persuasion techniques in multimodal content (memes). Our system combines pretrained multimodal models (CLIP) and chained classifiers. Also, we propose to enrich the data by a data augmentation technique. Our submission achieves a rank of 8/16 in terms of F1-micro and 9/16 with F1-macro on the test set.

## 1 Introduction

Online propaganda is potentially harmful to society, and the task of automated propaganda detection has been suggested to alleviate its risks (Martino et al., 2020b). In particular, providing a justification when performing propaganda detection is important for acceptability and application of the decisions. Previous challenges have focused on the detection of propaganda techniques (Martino et al., 2020a), based on news articles. However, many use cases do not solely involve text, but can also involve other modalities, notably images. Task 6 of SemEval-2021 proposes a shared task on the detection of persuasion techniques detection in memes, where both images and text are involved. Subtasks 1 and 2 deal with text in isolation, but we focus on subtask 3: visuolinguistic persuasion technique detection.

This article presents the system behind our submission for subtask 3 (Dimitrov et al., 2021). To handle this problem, we use a model containing three components: data augmentation, image and text feature extraction, and chain classifier components. First, given a paired image-text as the input, we paraphrase the text part using back-translation and pair it again with the corresponding image to enrich the data. Then, we extract visual and textual features using the CLIP (Radford et al., 2021) image encoder and text encoder, respectively. Finally, we use a chain classifier to model the relation between labels for the final prediction. Our proposed

method, named LIIR, has achieved a competitive performance with the best performing methods in the competition. Also, empirical results show that the augmentation approach is effective in improving the results.

The rest of the article is organized as follows. The next section reviews related works. Section 3 describes the methodology of our proposed method. We will discuss experiments and evaluation results in Sections 4 and 5, respectively. Finally, the last section contains the conclusion of our work.

## 2 Related work

This work is related to computational techniques for automated propaganda detection (Martino et al., 2020b) and is the continuation of a previous shared task (Martino et al., 2020a).

Taks 11 of SemEval-2020 proposes a more fine-grained analysis by also identifying the underlying techniques behind propaganda in news text, with annotations derived from previously proposed propaganda techniques typologies (Miller, 1939; Robinson, 2019).

This current iteration of the task tackles a more challenging domain, by including multimodal content, notably memes. The subtle interaction between text and image is an open challenge for state of the art multimodal models. For instance, the Hateful Memes challenge (Kiela et al., 2020) was recently proposed, as a binary task for detection of hateful content. The recent advances in pretraining of visuolinguistic representations (Chen et al., 2020) lead the model closer to human accuracy (Sandulescu, 2020).

More generally, propaganda detection is at the crossroad of many tasks, since it can be helped by many subtasks. Fact-checking (Aho and Ullman, 1972; Dale, 2017) can be involved with propaganda detection, alongside various social, emotional and discursive aspects (Sileo et al., 2019a),

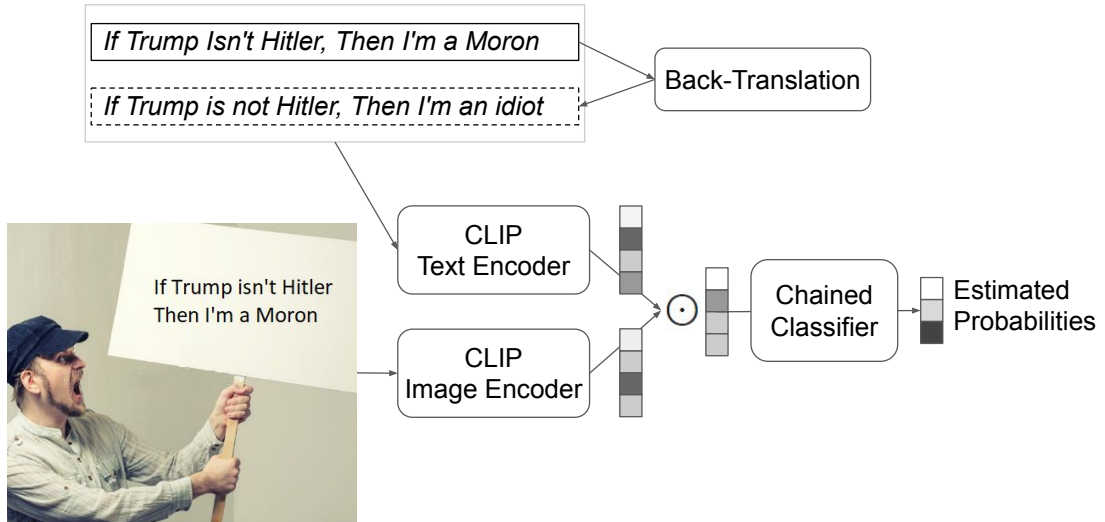


Figure 1: The overall architecture of our proposed model. For each example, use Back-Translation to derive augmentations of the text, and we compute persuasion techniques probabilities separately. Then, we average the estimated probabilities from augmented and original examples.

including offensive language detection (Pradhan et al., 2020; Ghadery and Moens, 2020) emotion analysis (Dolan, 2002), computational study of persuasiveness (Guerini et al., 2008; Carlile et al., 2018) and argumentation (Palau and Moens, 2009; Habernal and Gurevych, 2016).

### 3 Methodology

In this section, we introduce the design of our proposed method. The overall architecture of our method is depicted in figure 1. Our model consists of several components: a data augmentation component (Back-translation), a feature extraction component (CLIP), and a chained classifier. Details of each component are described in the following subsections.

#### 3.1 Augmentation Method

One of the challenges in this subtask is the low number of training data where the organizers have provided just 200 training samples. To enrich the training set we propose to use the back-translation technique (Sennrich et al., 2016) for paraphrasing a given sentence by translating it to a specific target language and translating back to the original language. To this end, we use four translation models, English-to-German, German-to-English, English-to-Russian, and Russian-to-English provided by (Ng et al., 2019). Therefore, for each training sentence, we obtain two paraphrased version of it. In

the test time, we average the probability distributions over the original and paraphrased sentence-image pairs.

#### 3.2 Feature Extraction

Our system isProbabilities of a combination of pre-trained visuolinguistic and linguistic models.

We use CLIP (Radford et al., 2021) as a pre-trained visuolinguistic model. CLIP provides an image encoder  $f_i$  and a text encoder  $f_t$ . They were pretrained on a prediction of matching image/text pairs. The training objective incentivizes high values of  $f_i(I) \cdot f_t(T)$  if  $I$  and  $T$  are matching in the training corpus, and low values of they are not matching<sup>1</sup>. Instead of using a dot product, we create features with element-wise product  $f_i(I) \odot f_t(T)$  of image and text encoding. This enables aspect-based representations of the matching between image and text. We experimented with other compositions (Sileo et al., 2019b) which did not lead to significant improvement.

We then use a classifier  $C$  on top of  $f_i(I) \odot f_t(T)$  to predict the labels.

#### 3.3 Chained Classifier

In this task, we are dealing with a multilabel classification problem, which means we need to predict a subset of labels for a given paired image-text sample as the input. We noticed that label

<sup>1</sup>They assign each image to the text associated to other images in the current batch to generate negative examples

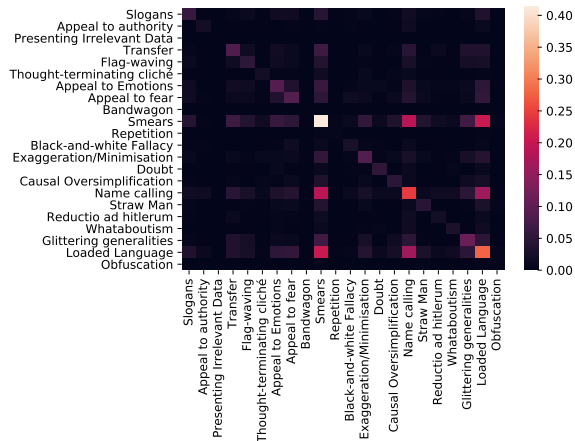


Figure 2: Probabilities of label co-occurrence in the training set. Some label pairs, for instance (SMEARS and LOADED LANGUAGE) are frequently associated.

co-occurrences were not uniformly distributed, as shown in figure 2. To further address the data sparsity, we use another inductive bias at the classifier-level with a chained classifier (Read et al., 2009) using scikit-learn implementation (Pedregosa et al., 2011).

Instead of considering each classification task independently, a chained classifier begins with the training of one classifier for each of the  $L$  labels. But we also sequentially train  $L$  other classifier instances thereafter, each of them using the outputs of the previous classifier as input. This allows our model to model the correlations between labels. We use a Logistic Regression with default parameter as our base classifier.

Our chain classifier uses combined image and text features as the input. We transfer the predicted probabilities of the classifier via the sigmoid activation function to make the probability values more discriminating (Ghadery et al., 2018). Then we apply thresholding on the  $L$  labels probabilities since the task requires a discrete set of labels as output. We predict a label when the associated probability is above a given threshold. We optimize the threshold on the validation set by a simple grid search using values between 0.0 and 0.9 with a step of 0.005.

## 4 Experiments

### 4.1 Datasets

We use the dataset provided by SemEval-2021 organizers for task 6. The dataset consists of 687(290) samples as the training set, 63 samples as the dev set, and 200 samples as the test set. Each sample

Label	Count
Smears	199
Loaded Language	134
Name calling/Labeling	118
Glittering generalities (Virtue)	54
Appeal to (Strong) Emotions	43
Appeal to fear/prejudice	42
Exaggeration/Minimisation	42
Transfer	41
Slogans	28
Doubt	25
Flag-waving	24
Causal Oversimplification	22
Misrepresentation of Someone’s Position	21
Whataboutism	14
Black-and-white Fallacy/Dictatorship	13
Thought-terminating cliché	10
Reductio ad hitlerum	10
Appeal to authority	10
Repetition	3
Obfuscation, Intentional vagueness, Confusion	3
Bandwagon	1
Presenting Irrelevant Data (Red Herring)	1

Table 1: Labels of persuasion techniques with associated counts in the training set

is an image and its corresponding text. We use 10% of the training set as the validation set for hyperparameter tuning.

## 5 Evaluation and Results

### 5.1 Results

In this section, we present the results obtained by our model on the test sets for Subtask 3. Table 2 shows the results obtained by the submitted final model on the test set. All the results are provided in terms of macro-F1 and Micro-F1. Furthermore, we provide the results obtained by the random baseline, the best performing method in the competition, and median result for the sake of comparison. Note that, we used the first released training set at the time of final submission which contained just 290 training samples. Therefore, we also provide results obtained by our model after using all the provided 687 training samples. Results show that LIIR has achieved a good performance compared to the majority class baseline and the median result which demonstrates that our model can effectively identify persuasion techniques in text and images. Also, we can observe LIIR has achieved a competitive performance compared to the best result obtained by the best team in the competition when it uses all the training samples.

System	Macro-F1	Micro-F1
Majority class	0.05152	0.07062
Median	0.18842	0.4896
LIIR(290 examples)	0.18807	0.49835
LIIR(687 examples)	0.21796	0.51122
Best system	<b>0.27315</b>	<b>0.58109</b>

Table 2: The results obtained by LIIR compared to the baselines on the Test set for Subtask 3. Numbers in parentheses show the total number of train samples used by our model.

## 5.2 Ablation Analysis

In this part, we provide an ablation study on the effect of different components of our proposed method on the dev set. First, we show the effect of using just visual features, just textual features, and both. Furthermore, we examine how well the final results of our model was influenced by the augmentation method. Table 3 shows the ablation study on the effect of using different features. The first observation is that image features contain more information compared to the textual features. Also, we can observe that the best Micro-F1 score is obtained when we combine both visual and textual features. These results show the effectiveness of our method in making use of both visual and textual information.

System	Macro-F1	Micro-F1
LIIR – textual features	0.32275	0.53237
LIIR – visual features	<b>0.33347</b>	0.52954
LIIR	0.29972	<b>0.58312</b>

Table 3: Ablation analysis for the effect of using different features by our model on the dev set.

In Table 4, the effect of the augmentation technique is shown. As the results show, the augmentation approach is quite effective in improving the model performance by a high margin.

System	Macro-F1	Micro-F1
LIIR w/o Augmentation	0.25090	0.54952
LIIR w Augmentation	<b>0.29972</b>	<b>0.58312</b>

Table 4: Ablation analysis for the effect of augmentation method on the dev set.

## 6 Negative Results

We also tried to use CLIP as a zero-shot classifier for propaganda technique detection. To do so, we constructed prompts such as :

(1) *This image is committing [LABEL] fallacy.*

or

(2) *Saying that [TEXT] is [LABEL] fallacy.*

For each input image/text, we generated a prompt for each labels, and used CLIP to estimate the estimate an affinity score between the prompt and the image. CLIP is designed to predict relatedness between the input image and text, and we expected that an input text mentioning the relevant propaganda technique should be associated with higher probabilities that the others.

However, this method did not seem to perform better than chance. This suggests that propaganda detection technique task might be too abstract for CLIP in zero-shot settings.

## 7 Conclusion

We described our submission for the shared task of multimodal propaganda technique detection at SemEval-2021. Our system performances that are competitive with other systems even though we used a simple architecture with no ensemble, by leveraging non-supervised learning. We believe that further work on zero-shot learning would be a valuable way to improve propaganda detection techniques for the least frequent labels.

## 8 Acknowledgments

This research was funded by the CELSA project from the KU Leuven with grant number CELSA/19/018.

## References

- Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation and Compiling*, volume 1. Prentice-Hall, Englewood Cliffs, NJ.
- Winston Carlile, Nishant Gurrupadi, Zixuan Ke, and Vincent Ng. 2018. [Give me more feedback: Annotating argument persuasiveness and related attributes in student essays](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 621–631, Melbourne, Australia. Association for Computational Linguistics.
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. [Uniter: Universal image-text representation learning](#). In *ECCV*.

- Robert Dale. 2017. NLP in a post-truth world. *Natural Language Engineering*, 23(2):319–324.
- Dimiter Dimitrov, Giovanni Da San Martino, Hamed Firooz, Fabrizio Silvestri, Preslav Nakov, Shaden Shaar, Firoj Alam, and Bishr Bin Ali. 2021. SemEval-2021 task 6: Detection of persuasion techniques in texts and images. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*. International Committee for Computational Linguistics.
- Raymond J Dolan. 2002. Emotion, cognition, and behavior. *Science*, 298(5596):1191–1194.
- Erfan Ghadery and Marie-Francine Moens. 2020. Liir at semeval-2020 task 12: A cross-lingual augmentation approach for multilingual offensive language identification. *arXiv preprint arXiv:2005.03695*.
- Erfan Ghadery, Sajad Movahedi, Hesham Faili, and Azadeh Shakery. 2018. An unsupervised approach for aspect category detection using soft cosine similarity measure. *arXiv preprint arXiv:1812.03361*.
- Marco Guerini, Carlo Strapparava, and Oliviero Stock. 2008. [Resources for persuasion](#). In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).
- Ivan Habernal and Iryna Gurevych. 2016. What makes a convincing argument? empirical analysis and detecting attributes of convincingness in web argumentation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1214–1223.
- Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. 2020. [The hateful memes challenge: Detecting hate speech in multimodal memes](#).
- G. Da San Martino, A. Barrón-Cedeño, H. Wachsmuth, R. Petrov, and P. Nakov. 2020a. [Semeval-2020 task 11: Detection of propaganda techniques in news articles](#).
- Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020b. A survey on computational propaganda detection. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4826–4832. International Joint Conferences on Artificial Intelligence Organization. Survey track.
- C.R. Miller. 1939. [How to Detect and Analyze Propaganda ....: An Address Delivered at Town Hall, Monday, February 20, 1939](#). A Town Hall pamphlet. Town Hall, Incorporated.
- Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. [Facebook FAIR's WMT19 news translation task submission](#). pages 314–319.
- Raquel Mochales Palau and Marie-Francine Moens. 2009. [Argumentation mining: The detection, classification and structure of arguments in text](#). In *Proceedings of the 12th International Conference on Artificial Intelligence and Law, ICAIL '09*, page 98–107, New York, NY, USA. Association for Computing Machinery.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Rahul Pradhan, Ankur Chaturvedi, Aprna Tripathi, and Dilip Kumar Sharma. 2020. A review on offensive language detection. In *Advances in Data and Information Sciences*, pages 433–439. Springer.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *Image*, 2:T2.
- Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 254–269. Springer.
- Robert C Robinson. 2019. A rulebook for arguments, by a. weston. *Teaching Philosophy*, 42(4):425–428.
- Vlad Sandulescu. 2020. [Detecting hateful memes using a multimodal deep ensemble](#). *CoRR*, abs/2012.13235.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Improving neural machine translation models with monolingual data](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.
- Damien Sileo, Tim Van de Cruys andmain Camille Pradel, and Philippe Muller. 2019a. [Discourse-based evaluation of language understanding](#). *CoRR*, abs/1907.08672.
- Damien Sileo, Tim Van De Cruys, Camille Pradel, and Philippe Muller. 2019b. [Composition of sentence embeddings: Lessons from statistical relational learning](#). In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (\*SEM 2019)*, pages 33–43, Minneapolis, Minnesota. Association for Computational Linguistics.

# AIMH at SemEval-2021 Task 6: Multimodal Classification Using an Ensemble of Transformer Models

Nicola Messina   Fabrizio Falchi   Claudio Gennaro   Giuseppe Amato  
ISTI-CNR, via G. Moruzzi 1, 56124 Pisa, Italy  
{name.surname}@isti.cnr.it

## Abstract

This paper describes the system used by the AIMH Team to approach the SemEval Task 6. We propose an approach that relies on an architecture based on the transformer model to process multimodal content (text and images) in memes. Our architecture, called DVTT (Double Visual Textual Transformer), approaches Subtasks 1 and 3 of Task 6 as multi-label classification problems, where the text and/or images of the meme are processed, and the probabilities of the presence of each possible persuasion technique are returned as a result. DVTT uses two complete networks of transformers that work on text and images that are mutually conditioned. One of the two modalities acts as the main one and the second one intervenes to enrich the first one, thus obtaining two distinct ways of operation. The two transformers outputs are merged by averaging the inferred probabilities for each possible label, and the overall network is trained end-to-end with a binary cross-entropy loss.

## 1 Introduction

Social networks play a critical role in our society. Nowadays, most of the ideas, thoughts, and political beliefs are shared through the internet using social platforms like Twitter, Facebook, or Instagram. Although these online services enable information to be spread efficiently and effectively, it is non-trivial to understand if the shared contents are free of subtle meanings altering people’s judgment abilities.

Among all the types of content living in a social network, memes acquire a significant role. Memes are small yet effective units of information able to spread cultural ideas, symbols, or practices and usually exist under the form of pictures, possibly with overlaid text. Memes are created so that they can propagate rapidly and reach a large number of users; for this reason, they are one of the most

popular types of content used in an online disinformation campaign, influencing the users through several rhetorical and psychological techniques, such as causal oversimplification, name-calling, or smear. The automatic detection of these memes and the disinformation techniques they are possibly employing is a challenging yet crucial task for the proper management of a social network.

In the last few years, machine learning and deep learning have defined remarkable milestones in automatic content extraction and reasoning from multimedia data. All these breakthroughs acquire a fundamental role in large-scale analysis of multimedia content from social networks.

In this work, we tackle the problem of recognizing which kind of disinformation technique is used to forge memes for a disinformation campaign. In particular, we propose an architecture based on the well-established transformer architecture model (Vaswani et al., 2017) for processing both the textual and visual inputs from the meme. This architecture, which we call DVTT (Double Visual Textual Transformer), comprises two full transformer networks working respectively on images and texts; however, each of these transformers is conditioned on the other modality. We consider this task as a multi-label classification problem, where text and/or images from the meme are processed, and probabilities of presence of each possible persuasion technique are returned as a result.

In this paper, we tackle subtasks 1 and 3 of the *SemEval 2021 Task 6* challenge<sup>1</sup> (Dimitrov et al., 2021). Subtask 1 consists of identifying which of 20 possible persuasion techniques are used in it given only the textual content; subtask 3 is very similar to subtask 1, but both textual and visual contents of the meme are used, and there are 22 possible persuasion techniques. Our proposed models

<sup>1</sup><https://propaganda.math.unipd.it/semEval2021task6/index.html>



could reach the 5th position for subtask 1 and the 4th position for subtask 3 on the publicly available leaderboard. The code for replicating our results is available on GitHub<sup>2</sup>.

## 2 Background

Recently, machine learning, and deep learning in particular, defined astonishing milestones in automatic content extraction and reasoning from multimedia data. In particular, concerning joint visual and textual analysis, many state-of-the-art approaches succeeded in tasks like visual question answering (Hu et al., 2017; Anderson et al., 2018; Teney et al., 2017), image captioning (Zhou et al., 2020; Rennie et al., 2017; Huang et al., 2019; Cornia et al., 2019), and image-text matching (Chen et al., 2019; Lu et al., 2019; Faghri et al., 2018; Lee et al., 2018; Messina et al., 2020), often using structured reasoning using graph networks and graph convolutions.

In the last few years, a graph-network related model, the transformer network (Vaswani et al., 2017), acquired increasing attention on the joint processing of images and texts. Many works, inspired by the BERT model (Devlin et al., 2019), obtained remarkable results on word region alignments, visual-question answering, and image-text matching using transformer encoders (Lu et al., 2019; Qi et al., 2020; Su et al., 2020).

Recently, the authors in (Carion et al., 2020) used the full transformer stack to construct a powerful object detector, demonstrating these models' potential in pure visual contexts.

Given the enormous flexibility of the transformer architecture, in this work, we consider images and texts respectively as sets and sequences of vectors, and we ask the transformer to process them to produce probabilities of presence of each possible persuasion technique.

## 3 System Overview

In this section, we first give a brief overview of the Transformer architecture on which our proposal is based; then, we present in detail our system proposals for solving subtasks 1 and 3.

### 3.1 Review of the Transformer Architecture

In the original transformer formulation for language translation, the source sequence is processed

<sup>2</sup><https://github.com/mesnico/MemePersuasionDetection>

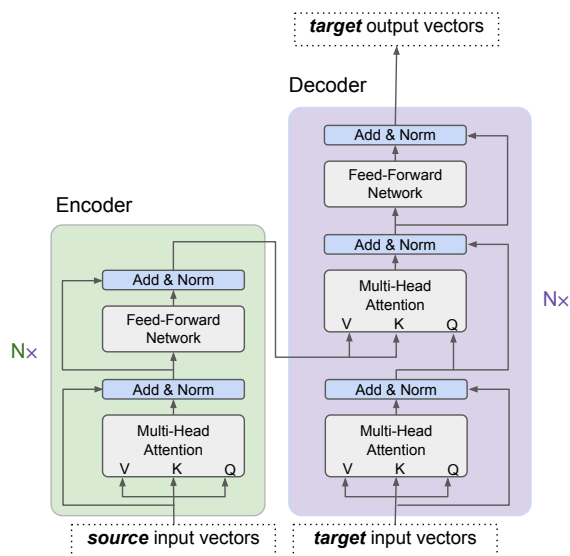


Figure 1: The transformer network. Encoder and Decoder modules are replicated N times.

using the *transformer encoder* model, which creates a suitable set of contextualized vectors encoding the input sequence. Using the representations created by the encoder, the *transformer decoder* module is trained to predict the words for the target sentence one at a time. During the decoding process, the decoder is conditioned, at each time step, by the vectors generated by the encoder.

Both the encoder and the decoder modules leverage the power of the multi-head attention mechanism. This mechanism transforms every word representation from a target sentence to a new representation space conditioned on the words from a source sentence.

The multi-head attention associates to the source sequence vectors  $\{s_i\}$  a key  $K_i$  and to the target vectors  $\{t_j\}$  a query  $Q_j$  and a value  $V_j$ ; the target values are transformed using the scaled dot-product attention as follows:

$$\text{Att}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V. \quad (1)$$

This is the core of the multi-head attention mechanism, which is the fundamental building block in the transformer architecture for both the encoder and the decoder modules (Figure 1).

### 3.2 A Transformer Encoder Baseline

Although the transformer architecture was originally designed to handle sentences (sequences of words), the model in itself can effectively process an arbitrary set of vectors possibly coming from

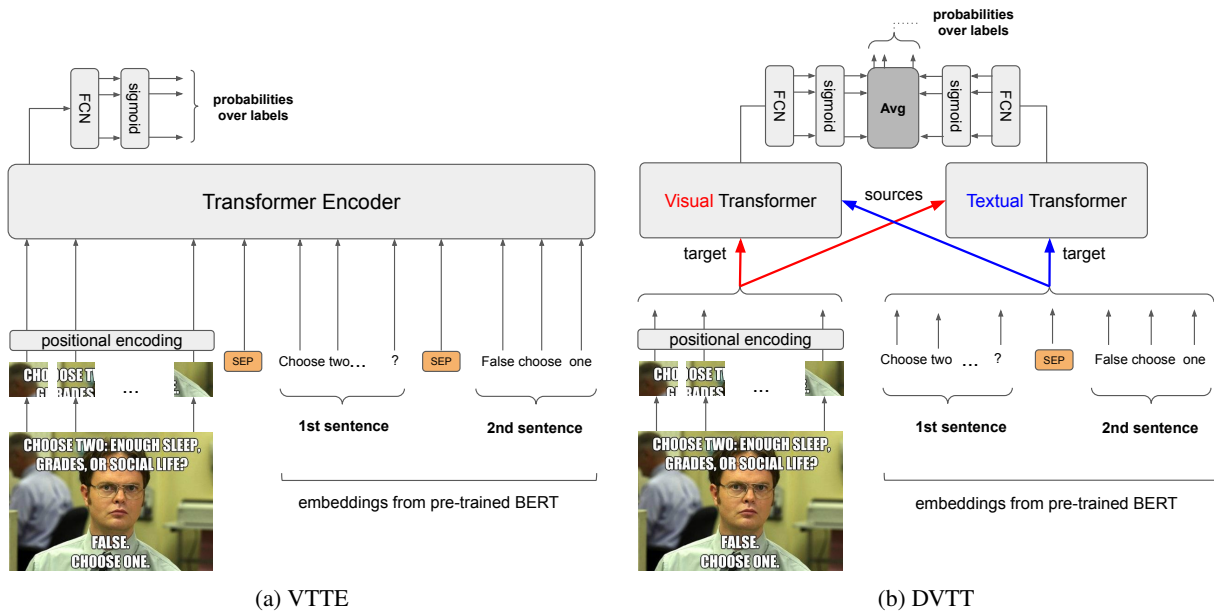


Figure 2: The proposed architectures: (a) the Visual-Textual Transformer Encoder model (VTTE) used as a baseline, and (b) the novel Double Visual Textual Transformer model (DVTT). The shown meme is taken from <https://engineermemes.blogspot.com>, and it is licensed under the Creative Common license.

other modalities (e.g., different chunks of an image).

For this reason, many works (Lu et al., 2019; Qi et al., 2020; Su et al., 2020) recently proposed architectures based on the transformer encoder model to jointly reason on images and texts for solving tasks like visual question answering or image-text matching.

Inspired by these works, we defined a baseline for inferring probabilities over the possible persuasion techniques by feeding images and texts to a transformer encoder, using the first output token as the input to the multi-label classifier head. The transformer encoder visual and textual input features are pre-extracted respectively from a CNN and a pre-trained BERT model, as explained in Section 4. Like in BERT, where different sentences are encoded by separating them using a special token, the textual and the visual inputs are separated by the SEP embedding. An overview of this approach is presented in Figure 2a. We refer to this baseline as *VTTE* (Visual-Textual Transformer Encoder).

### 3.3 Double Visual-Textual Transformer

In this work, instead, we propose an architecture that can exploit the full Transformer architecture to jointly reason on visual and texts and producing label probabilities as output. We call this model *DVTT* (Double Visual Textual Transformer), and it is outlined in Figure 2b. DVTT is composed of two

different transformer networks able to process visual and textual inputs concurrently; the important aspect of DVTT is that each transformer is conditioned on the other modality so that it is possible for the whole architecture to jointly reason on the two modalities following two different paths: in the first, the text is the key aspect, and images integrate the reasoning performed on the text; conversely, in the second, the images are the primary modality and the text intervene to enrich the visual features.

For each of the two transformers, the final head is a multi-classification head constructed on the first token of the output sequence. In particular, a linear layer outputs the logits over each possible persuasion technique, and the final softmax operator converts logits into probabilities, exactly like in the VTTE baseline model. The two transformers outputs are merged by averaging the inferred probabilities for each possible label, and the overall network is trained end-to-end with a binary cross-entropy loss.

## 4 Experiments

We used the data provided by the *SemEval 2021 Task 6* challenge organizers to train and validate our model. Although we mainly concentrated on subtask 3 (images + texts), we also tackled subtask 1, which is essentially equivalent to subtask 3, except that only the text is available.

**Dataset** The provided dataset comprises 687 memes for training, 63 memes for validating on the so-called development set, and 200 memes for the final testing. All the memes carry textual captions written in English. Note that, in the end, we were allowed to use the annotations for the development set, so we had at our disposal a total of 750 annotated memes to use for the training and validation phases. The annotations consist of a list of persuasion techniques for every meme. In subtask 1 there are 20 possible persuasion techniques and 22 in subtask 3.

**Metrics** The official metrics for computing the model performance are the Micro- $F_1$  and Macro- $F_1$  scores;

The  $F_1$ -score is defined as the harmonic mean of precision and recall:

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \quad (2)$$

The  $F_1$ -score gives values in the interval  $[0, 1]$ , hence it is often a good way of summarizing the performance of binary classifiers.

The difference between Micro- $F_1$  and Macro- $F_1$  scores lies in the way *precision* and *recall* are computed: in Micro- $F_1$ , they are computed from all the true positives, false positives, and false negatives over all the labels; for this reason, Micro- $F_1$  gives each sample the same weight, thus giving more emphasis to the most frequent labels. On the other hand, Macro- $F_1$  is computed as the mean value among the  $F_1$ -scores computed on the different labels:  $\text{Macro-}F_1 = \frac{1}{N} \sum_1^N F_1^i$ , where  $N$  is the number of labels and  $F_1^i$  is the  $F_1$ -score computed among the samples having label  $i$ . In this case, all the classes contribute equally regardless of how often they appear in the dataset.

**Model Setup** For subtask 3, we used the proposed DVTT model (Figure 2b). We used a learning rate of  $5 \cdot 10^{-5}$  and a batch size of 8. We trained the models for 40 epochs in all the experiments, decreasing the learning rate after 30 epochs to  $5 \cdot 10^{-6}$ . The transformer is composed of 4 encoder layers and 4 decoder layers, with 1024-dimensional feed-forward networks for producing queries, keys, and values.

As a baseline for subtask 3, we used the VTTE architecture (shown in Figure 2a), composed of a 4-layer transformer encoder module, with a multi-label classification head on top, exactly like the one

in DVTT. For subtask 1, instead, we used the VTTE architecture (Figure 2a) with the same setup used for the subtask 3 baseline, except that the visual input is not fed to the network.

**Features Extraction** For all the conducted experiments, we obtained suitable visual and textual features from pre-trained state-of-the-art networks. Concerning images, we re-scaled them to  $256 \times 256$ , and we took a  $224 \times 224$  crop (a random crop during training and a center crop during inference). We also normalized the images using the pixels mean and standard deviation computed on the whole dataset.

In order to input an image to the transformer, we had to encode it as a set of features. We used a ResNet50 pre-trained on image classification, as it is characterized by a good performance at low computational costs compared to deeper backbones; we down-sampled the features maps from the last convolutional layer to a  $7 \times 7$  spatial grid of 2048-dimensional features. The resulting flattened 49 visual features were then augmented with their spatial positions by appending the normalized coordinates of the chunk to the 2048-dimensional visual feature. Another possibility consisted of using visual features extracted from state-of-the-art object detectors, like Faster-RCNN. However, images carried in memes are not homogeneous: they show possible stacked scenes and overlaid text, making it very difficult for an object detector to identify the most critical regions.

Concerning text processing, we used a pre-trained BERT model (Devlin et al., 2019) for extracting word embeddings. BERT embeddings are trained on some generic language processing tasks such as sentence prediction or sentence classification and demonstrated state-of-the-art results in many downstream tasks. Every meme can carry one or more sentences, encoded in the same string and separated by “\n\n”. For this reason, during the string tokenization phase, we simply replaced “\n\n” with the SEP token. In the basic DVTT model, we trained only the transformer models, leaving the feature extractor fixed. In the Experiments section, we also report the results from a fine-tuning of the feature extractors.

**Validation** The test-set annotations were hidden to the participants, so the model should be validated using a split of the available annotated data. Given that the available annotated memes are relatively

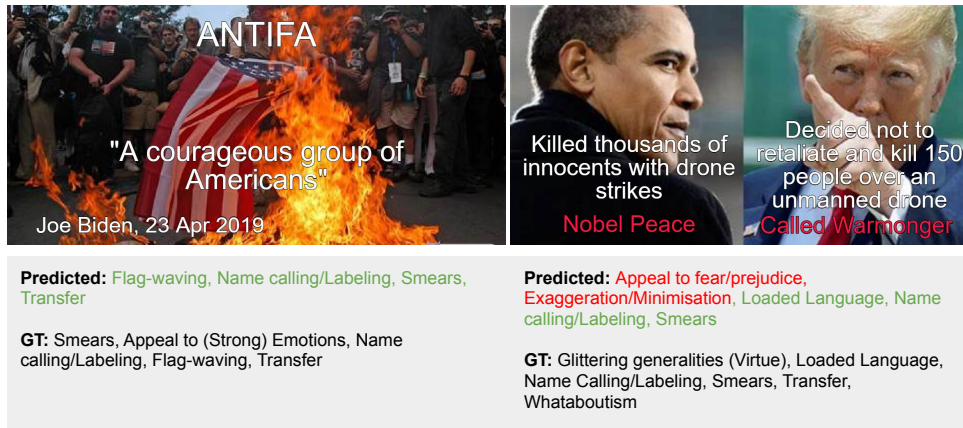


Figure 3: Example of predictions from the DVTT model for subtask 3. In green, the true positives labels; in red, the false positives labels. Images obey to the Creative Common license and they are searched on the Bing image-search engine using "free to modify, share and use" license filtering.

Model	Macro- $F_1$	Micro- $F_1$
VTTE (Baseline)	0.327	0.596
DVTT	0.336	<b>0.601</b>
DVTT - Balanced	0.300	0.489
DVTT - Finetuned	<b>0.341</b>	0.592
DVTT - 2 layers	0.310	0.596
DVTT - 6 layers	0.325	0.583

Table 1: Ablation results on subtask 3

Model	Macro- $F_1$	Micro- $F_1$
VTTE	0.372	0.566
VTTE - Balanced	0.361	0.490
VTTE - Finetuned	0.386	<b>0.581</b>
VTTE - 2 layers	0.365	0.565
VTTE - 6 layers	<b>0.389</b>	0.569

Table 2: Ablation results on subtask 1. Note that VTTE in this case does not receive the images in input.

few, we validated our model using cross-validation. In particular, we split the training data into six different folds, training six different models by using five out of six data folds and validating them using the remaining fold. We selected the model having the best sum of Micro- $F_1$  and Macro- $F_1$  scores on the validation fold. All the performance measures reported in the Results section are an average of the metrics from this 6-fold validation procedure.

For participating in the final competition on the test set, we prepared an ensemble model composed of all the six trained models, and we produced the final probabilities by soft-voting.

We used a final binary-classification threshold of 0.3 over the label probabilities.

## 5 Results

Concerning subtask 3, we studied the performance of our DVTT model by comparing the  $F_1$ -scores against the VTTE baseline; furthermore, we tried also to train the model using a balanced sampling of the labels and to fine-tune the feature extractors (BERT and the ResNet-50), using a learning rate of 1/10 with respect to the one used for training the transformer models. Using a lower learning rate during the fine-tuning process is a common procedure to avoid model overfitting. We also report the results of slightly different variants of the DVTT model obtained by increasing and decreasing the number of the transformer's encoder / decoder layers: the base architecture contains four layers; we also experimented with two and six. The results of these experiments are reported in Table 1.

For subtask 1, instead, we used the VTTE model without visual input, trying out the same experiments performed for subtask 3. In this case, when varying the number of layers, we only considered the transformer encoder ones (there is no decoder in the VTTE model). The ablation results on subtask 1 are reported in Table 2.

## 6 Discussion

Looking at the subtask 3 results in Table 1, we can notice that the proposed DVTT model can achieve slightly better results than the VTTE baseline. In particular, the DVTT with fine-tuned BERT and ResNet50 modules achieve the best results on the Macro- $F_1$  metric. Also, the choice of using four encoder and decoder layers seems to lead to the best compromise on both the metrics. Concerning the results of subtask 1 in Table 2, fine-tuning the BERT model is even in this case a good choice. Fine-tuning the feature extractors, in fact, enables the model to slightly adjust the weights of the backbones pre-trained on generic tasks to align them to the specific domain.

Figure 3 reports some examples of predictions from our model for subtask 3. We evidenced in green the true positives and in red the false positives. The model can correctly identify most of the persuasion techniques. However, there are cases where it is probably necessary to access more contextual information to solve the most complex labels. For example, in the second meme from the left, the model outputs the label *Exaggeration/Minimisation* probably due to the presence of vague quantities (*Killed thousands of innocents*). It would be necessary to access external data to effectively reason on the complex common sense and historical facts hidden behind these complex memes.

## 7 Conclusions

In this work, we proposed transformer-based models for tackling subtasks 1 and 3 of the SemEval-2021 Task 6, concerning the identification of persuasion techniques in memes. In particular, for subtask 3 which involves both images and texts from the meme, we proposed a Double Visual Textual Transformer (DVTT) model. This model uses the full power of the transformer architecture; it demonstrated better results than the baseline, which is composed of a single transformer encoder module fed with both images and text. On the public leaderboard, we reached 4th place on subtask 3. Using the baseline model, which can process text alone without images, we also tackled subtask 1, reaching 5th place on the public leaderboard.

In the future, we plan to improve our visual feature extraction pipeline, using face expression detection and classification and possibly employing ad-hoc trained object detectors suitable for meme

images. Also, it would be interesting to study the effective reasoning abilities of the proposed models, by leveraging the attention mechanisms of the transformer, possibly integrating the data with a knowledge base of historical facts that helps to create a more suitable context.

## Acknowledgments

This work was partially supported by “Intelligenza Artificiale per il Monitoraggio Visuale dei Siti Culturali” (AI4CHSites) CNR4C program, CUP B15J19001040004, by the AI4EU project, funded by the EC (H2020 - Contract n. 825619), and AI4Media under GA 951911.

## References

- Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. 2018. Bottom-up and top-down attention for image captioning and visual question answering. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086.
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer.
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019. Uniter: Learning universal image-text representations. *arXiv preprint arXiv:1909.11740*.
- Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. 2019. Show, control and tell: A framework for generating controllable and grounded captions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 8307–8316.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics.
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Feroz Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *In Proceedings of the 15th International Workshop on Semantic Evaluation*.
- Fartash Faghri, David J. Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: improving visual-semantic embeddings with hard negatives. In *BMVC 2018*, page 12. BMVA Press.

- Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proc. of the IEEE International Conference on Computer Vision*, pages 804–813.
- Lun Huang, Wenmin Wang, Jie Chen, and Xiao-Yong Wei. 2019. Attention on attention for image captioning. In *Proc. of the IEEE International Conference on Computer Vision*, pages 4634–4643.
- Kuang-Huei Lee, Xi Chen, Gang Hua, Houdong Hu, and Xiaodong He. 2018. Stacked cross attention for image-text matching. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 201–216.
- Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems*, pages 13–23.
- Nicola Messina, Fabrizio Falchi, Andrea Esuli, and Giuseppe Amato. 2020. Transformer reasoning network for image-text matching and retrieval. In *International Conference on Pattern Recognition (ICPR) 2020 (Accepted)*.
- Di Qi, Lin Su, Jia Song, Edward Cui, Taroon Bharti, and Arun Sacheti. 2020. Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data. *arXiv preprint arXiv:2001.07966*.
- Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.
- Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. Vi-bert: Pre-training of generic visual-linguistic representations. In *International Conference on Learning Representations*.
- Damien Teney, Lingqiao Liu, and Anton van Den Hengel. 2017. Graph-structured representations for visual question answering. In *Proc. of the IEEE conference on computer vision and pattern recognition*, pages 1–9.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Luowei Zhou, Hamid Palangi, Lei Zhang, Houdong Hu, Jason J Corso, and Jianfeng Gao. 2020. Unified vision-language pre-training for image captioning and vqa. In *AAAI*, pages 13041–13049.

# HOMADOS at SemEval-2021 Task 6: Multi-Task Learning for Propaganda Detection

Konrad Kaczyński<sup>1,2</sup> and Piotr Przybyła<sup>1</sup>

<sup>1</sup>Institute of Computer Science, Polish Academy of Sciences

<sup>2</sup>Faculty of Economic Sciences, University of Warsaw

{konrad.kaczynski, piotr.przybyla}@ipipan.waw.pl

## Abstract

Among the tasks motivated by the proliferation of misinformation, propaganda detection is particularly challenging due to the deficit of fine-grained manual annotations required to train machine learning models. Here we show how data from other related tasks, including credibility assessment, can be leveraged in multi-task learning (MTL) framework to accelerate the training process. To that end, we design a BERT-based model with multiple output layers, train it in several MTL scenarios and perform evaluation against the SemEval gold standard.

## 1 Introduction

Fine-grained propaganda detection is a new approach to tackling online misinformation, highlighting instances of propaganda techniques on the word level. These techniques are used in textual communication in order to encourage certain beliefs, but instead of straightforward presentation of arguments, they rely on psychological manipulation, logical fallacies or emotion elicitation.

There are general-purpose natural language processing (NLP) methods that could be used for automatic detection of such text fragments. The challenge here is that they require large amounts of training data, which are laborious to produce. However, propaganda techniques are often related to other misinformation challenges, for which large datasets do exist, e.g. credibility assessment or fake news detection.

In the present study we aim to investigate how this connection can be used in the multi-task learning (MTL) framework. We show how the performance of multi-label token-level propaganda detection within shared task 6 at SemEval-2021 can be improved by building neural architectures that are also trained to solve other tasks: single-label propaganda detection from SemEval-2020

and document-level credibility assessment based on a fake news corpus. We check different MTL scenarios (parallel and sequential) and show which aspects of the model benefit the most from this approach.

## 2 Problem Statement

We participate in SemEval-2021 Task 6 („Detection of Persuasion Techniques in Texts and Images”), subtask 2 (Dimitrov et al., 2021), where the goal is to identify all propaganda techniques within a provided fragment of text. Specifically, given a character sequence  $\langle c_0, c_1, \dots, c_N \rangle$ , we aim to produce a set of annotations  $\{(b_0, e_0, t_0), (b_1, e_1, t_1), \dots, (b_k, e_k, t_k)\}$ , where each triple consists of the character offsets of the span it covers ( $0 \leq b_i < e_i \leq N$ ) and an indication which one from the set of 20 known techniques is detected there ( $t_i \in T$ ). We can see it as a multi-label sequence classification task (Read et al., 2009), where each character (or token) can be assigned from 0 to 20 labels.

## 3 Related Work

Propaganda has been observed in text for a long time, but the problem of automatic detection of such techniques was posed just recently. Initially, a lack of word-level datasets confined the analysis to document-level classification, e.g. based on stylometric features (Rashkin et al., 2017; Barrón-Cedeño et al., 2019). Classification on the word level became possible with the dataset (Da San Martino et al., 2019b) released for the „Fine-Grained Propaganda Detection” shared task at the NLP4IF 2019 workshop (Da San Martino et al., 2019a). The corpus includes 550 news articles annotated with propaganda techniques on the word level. Among the submissions, the best performing models were based on word embeddings and pretrained lan-

guage models, such as BERT (Devlin et al., 2018). To tackle the data sparsity problem, participants employed various over-sampling methods or trained their models on auxiliary data. Similar objectives were pursued at SemEval 2020 Task 11, consisting of two subtasks: binary sequence tagging task and multi-class classification task. The majority of tasks’ participants based their solutions on the Transformers architecture (Vaswani et al., 2017) and word embeddings, combining them with other neural architectures (e.g. CNNs or LSTMs) or models such as CRF and logistic regression. Ensemble models were able to achieve satisfactory results when performing both tasks jointly.

## 4 Methods

### 4.1 Data Description

We make use of three datasets in English. In all the following approaches, the main focus is on the corpus released for SemEval-2021 Task 6 (Dimitrov et al., 2021) (S21). Additionally, we utilise the corpora from SemEval-2020 Task 11 (S20) (Da San Martino et al., 2020) and news credibility research (FN) (Przybyła, 2020).

S21 consists of text of 870 memes (607, 63 and 200 in the training, development and test subsets, respectively) annotated with 1550 spans a few words long (40 characters on average), each from one of 20 propaganda techniques. Most commonly occurring techniques are *Loaded language* (35%), *Name Calling/Labelling* (19%) and *Smears* (12%).

S20 corpus consists of 446 press articles (371 and 75 in the training and development subsets, respectively) annotated with 14 propaganda categories on a word level. Among the 7192 annotated spans, *Loaded language* (34%), *Name Calling/Labelling* (17%) and *Repetition* (12%) are most common categories. Given that very few spans overlap (8%), we represent the task as single-label classification by merging these spans according to their order in corresponding label files. Finally, we exclude sentences that do not contain any propaganda annotations.

To obtain the FN data, from the original corpus of 103,219 news articles classified as either credible or non-credible based on their source, we randomly select a sample of fifty thousand sentences with a binary credibility label.

### 4.2 Multi-Task Architecture

Figure 1 shows the architecture designed to fulfil the MTL objectives. A text document (usually one sentence) is first processed by BERT, resulting in 768-dimensional vectors:  $h_i$  for the  $i$ -th token and  $h_0$  for the whole document, using the [CLS] token. These vectors are processed by classification modules  $D_x$ , each consisting of a linear dense layer and a softmax activation function. Three types of such operations are considered:

- $d_0 = D_d(h_0)$ : document-level representation is used to produce 2-dimensional score vector ( $d_0$ ), indicating class probabilities in binary single-label classification,
- $s_i = D_s(h_i)$ : token-level representation is used to produce  $k$ -dimensional score vector ( $s_i$ ), indicating class probabilities in multi-class single-label classification,
- $m_i = D_m(h_i)$ : token-level representation is used to produce  $l \times 2$ -dimensional score vector ( $m_i$ ), indicating class probabilities in multi-class multi-label classification.

The following subsections describe several scenarios of using these three output types to improve the accuracy of propaganda detection.

### 4.3 Single Task

In the primary method we use BERT-Base-Uncased with the token-level multi-label classification layer, trained using only S21 data (SINGLE S21). The output for the  $i$ -token, denoted by  $m_i$ , is a  $20 \times 2$  matrix, in which the  $j$ -th row reflects the probability of the  $j$ -th propaganda technique being present in this token. If the token does not participate in any propaganda techniques, the first column of the matrix will be filled with ones and the second one with zeros. Since the S21 corpus is annotated at the character level, during preprocessing we map the initial annotation into tokens obtained via Word-Piece tokenisation.

### 4.4 Sequential Multi-Task Learning

In case of sequential MTL, the main training described in previous section is preceded with training for one of two auxiliary tasks:

- single-label classification task on S20 corpus (MULTI-S S20-S21).
- document-level classification task with FN corpus (MULTI-S FN-S21).



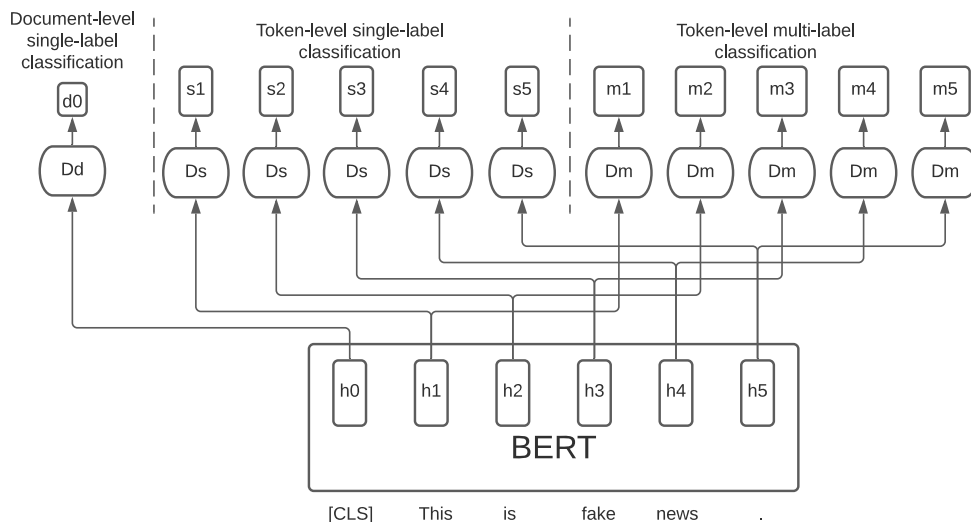


Figure 1: Multi-task architecture of our solution

For MULTI-S S20-S21, we involve the token-level single-label classification layer to produce 16-dimensional  $s_i$  vector. This allows to classify each token in S20 corpus into one of 16 categories (14 propaganda + non-propaganda + padding). MULTI-S FN-S21 uses the document-level single-label classification output ( $d_0$ ) layer for classifying sentences from FN corpus as coming from credible ( $d_0 = (0, 1)$ ) or non-credible ( $d_0 = (1, 0)$ ) articles.

For each model the learning procedure is the same: first, during an auxiliary task, only the additional classification layer is trained using cross-entropy loss and the auxiliary data. In the second phase, the training continues as a regular task on S21 data, as described in the previous section. Weights of all trainable variables are being updated in both phases.

#### 4.5 Parallel Multi-Task Learning

In the parallel MTL objective, the auxiliary task and the target task are learnt jointly. Similarly to sequential MTL, we devise two models, each consisting of BERT with two separate classification layers on top:

- single-label and multi-label classification on S20 and S21 corpora (MULTI-P S20-S21),
- document-level and multi-label classification tasks on FN and S21 corpora (MULTI-P FN-S21).

Every batch of data consists of sentences coming from both datasets: four sentences from S20 or FN and four sentences from S21 are sent through their corresponding classification layers to produce

outputs, and then count losses and update weights based on appropriate losses.

## 5 Evaluation

### 5.1 Experimental setup

We train our models according to multi-task scenarios, and use development subset of S21 to choose optimal number of training epochs of the final phase. The model trained up to this point is applied to test data to produce final predictions. In case of sequential MTL scenarios, this is preceded by training on additional corpora: on S20 for 10 epochs or on FN for 1 epoch. In case of parallel multi-task scenarios, the difference of training set sizes requires a special approach. For S20-S21, one epoch of training covers the whole S21 and 1/9 of S20. For FN-S21, we choose a balanced sub-sample of 18 thousand sentences and each training epoch covers the whole S21 and 1/30 of this sub-sample.

We use cross-entropy as the loss function, and compute it only for non-padding tokens. For all experiments we use a maximum sequence length of 210 tokens, Adam optimizer (Kingma and Ba, 2015) with the learning rate of  $3 \times 10^{-5}$  and batch size of four sentences. We use the L1 regularisation (Ng, 2004) with  $\alpha = 0.01$ . During fine-tuning of the model, weights of all trainable variables, including those in BERT, are being updated. During inference, we translate token-level labels back to character-level labels, including spaces and punctuation marks between adjacent tokens with identical labels. All experiments are conducted within the *TensorFlow* framework.

Approach	Dev			Test		
	F1	Precision	Recall	F1	Precision	Recall
SINGLE S21	0.5412	0.5798	0.5075	<b>0.4571</b>	0.4752	0.4403
MULTI-S S20-S21	0.5084	0.5181	0.4990	0.4444	0.4500	0.4390
MULTI-S FN-S21	0.4581	0.4836	0.4351	0.4185	0.4778	0.3723
MULTI-M S20-S21	<b>0.5455</b>	0.5747	0.5191	<u>0.4074</u>	<u>0.4121</u>	<u>0.4028</u>
MULTI-M FN-S21	0.5291	0.6429	0.4496	0.4381	0.5307	0.3730

Table 1: Propaganda detection performance on the development and test set for different evaluated approaches. The best F1 scores are highlighted. The run submitted to the shared task is underlined.

## 5.2 Evaluation measures

To evaluate our results we use character-level F1 measure prepared for the shared task (Dimitrov et al., 2021). It compares model’s results with the golden annotations, accounting for the imbalance of categories and partial overlaps between fragments with the same label.

## 6 Results

Table 1 shows the performance of the considered approaches on the development and test set. The highest F1 score on the development set was obtained by the MULTI-P S20-S21 model. Hence, this model was used to generate the predictions on the test set submitted to the shared task (underlined). However, we can see that the single task approach is not far behind on the development set and actually provides the best performance on the test set. The differences between approaches are relatively modest and no single model outperforms others on each set and metric. This is mostly due to the small size of the propaganda datasets. Specifically, choosing the approach and number of training epochs based on the development set, which contains just 63 documents, may lead to overfitting.

In order to better understand how the introduction of MTL influences the models, we perform additional experiments. Firstly, in Table 2 we show F1 score for the recognition of each technique in single task and sequential MTL scenarios using both auxiliary datasets. One could expect the usage of S20, annotated with a similar set of propaganda techniques, to improve performance for overlapping labels, but the data do not confirm this. For example, the performance for the relatively large (12.7%) *Smears* (Smr) category improves noticeably, even though it was not present in S20. At the same time, we see F1 drop in case of some techniques present in both datasets, such as *Appeal to authority* (AtA) or *Slogans* (Slg). Clearly, the

Technique	S21	M-S S20	M-S FN
AtA	0.6316	0.0000	0.7273
Atfp	0.0000	0.3966	0.0000
Bwf\D	0.6292	0.5824	0.0000
CO	0.0000	0.1667	0.0000
Dbt	0.0000	0.4578	0.1778
Ex-Min	0.4957	0.3221	0.5041
FW	0.3333	0.5397	0.0000
Gg	0.2222	0.0000	0.0000
LL	0.7038	0.6385	0.7135
NC-L	0.6136	0.6159	0.6830
Slg	0.3448	0.0000	0.3750
Smr	0.3839	0.5756	0.4743
Whbt	0.3830	0.0000	0.2222

Table 2: Per-technique F1 score on test set for different auxiliary datasets: S20 propaganda (S20) and fake news (FN) used in sequential multi-task scenario (techniques with no performance differences omitted for brevity).

language constructions covered by these labels in case of press articles and memes are too different to offer clear advantage of MTL. The relationship with fake news detection is even weaker, resulting in many techniques not being recognised.

Secondly, in Figure 2 we show how F1 on test set changes during training on S21 for the single task configuration and two scenarios based on S20 data: sequential and parallel. As expected, we see that pre-training allows our model to obtain good performance much faster, e.g. reaching F1=0.4 after 7 epochs instead of 14. But after longer training, the single-task approach catches up and beyond 20th epoch, when all version reach stable results, it outperforms the MTL variants.

## 7 Conclusion

In this work we explore how detection of propaganda techniques in text of memes can be facil-

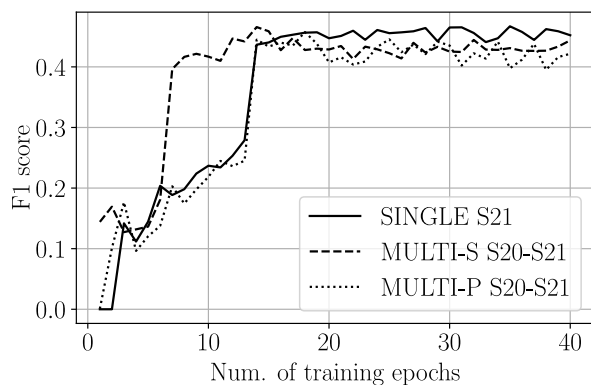


Figure 2: F1 scores during training for single task approach and multi-task learning using S20 data.

itated using external data in multi-task learning framework. The results show that the auxiliary tasks indeed influence the results, both in terms of accelerating the learning process and changing the set of recognised techniques. Nevertheless, these modifications do not offer clear advantages over the basic BERT-based solution.

We hypothesise this is because the link between main and auxiliary tasks is not strong enough to deliver benefits through multi-task learning. Additionally, propaganda is rarely a straightforward phenomenon and different techniques may require tailored approaches. We treat this effort as a preliminary study and aim to further investigate MTL’s relevance in detecting propaganda by extending the auxiliary tasks portfolio with corpora reflecting other related issues, such as hate speech or hyper-partisan language.

## Acknowledgements

This work was supported by the *Polish National Agency for Academic Exchange* through a *Polish Returns* grant number PPN/PPO/2018/1/00006.

## References

Alberto Barrón-Cedeño, Israa Jaradat, Giovanni Da San Martino, and Preslav Nakov. 2019. [Propy: Organizing the news based on their propagandistic content](#). *Information Processing and Management*, 56(5):1849–1864.

Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019a. [Findings of the NLP4IF-2019 Shared Task on Fine-Grained Propaganda Detection](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 162–170, Hong Kong, China.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. [SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, Barcelona (online).

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019b. [Fine-grained analysis of propaganda in news articles](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5636–5646, Hong Kong, China.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT 2019)*, pages 4171–4186.

Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. [Task 6 at SemEval-2021: Detection of Persuasion Techniques in Texts and Images](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation*, Bangkok, Thailand.

Diederik P. Kingma and Jimmy Lei Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.

Andrew Y. Ng. 2004. [Feature selection, L1 vs. L2 regularization, and rotational invariance](#). In *ICML ’04: Proceedings of the 21st International Conference on Machine Learning*.

Piotr Przybyła. 2020. [Capturing the Style of Fake News](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):490–497.

Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. [Truth of varying shades: Analyzing language in fake news and political fact-checking](#). In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 2931–2937.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2009. [Classifier chains for multi-label classification](#). In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5782 LNAI, pages 254–269.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009.

# 1213Li at SemEval-2021 Task 6: Detection of Propaganda with Multi-modal Attention and Pre-trained Models

Peiguang Li<sup>1,2,3,4</sup>, Xuan Li<sup>1,2,3,4,\*</sup>, Xian Sun<sup>1,2,3,4,†</sup>

<sup>1</sup> Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China

<sup>2</sup> Key Laboratory of Network Information System Technology (NIST), Aerospace Information Research Institute, Chinese Academy of Sciences, Beijing 100190, China

<sup>3</sup> University of Chinese Academy of Sciences, Beijing 100190, China

<sup>4</sup> School of Electronic, Electrical and Communication Engineering, University of Chinese Academy of Sciences, Beijing 100190, China

{lpeiguang17, lixuan173}@mailsucas.ac.cn, sunxian@aircas.ac.cn

## Abstract

This paper presents the solution proposed by the 1213Li team for subtask 3 in SemEval-2021 Task 6: identifying the multiple persuasion techniques used in the multi-modal content of the meme. We explored various approaches in feature extraction and the detection of persuasion labels. Our final model employs pre-trained models including RoBERTa and ResNet-50 as a feature extractor for texts and images, respectively, and adopts a label embedding layer with multi-modal attention mechanism to measure the similarity of labels with the multi-modal information and fuse features for label prediction. Our proposed method outperforms the provided baseline method and achieves 3rd out of 16 participants with 0.54860/0.22830 for Micro/Macro F1 scores.

## 1 Introduction

The development of the Internet and Information Technology promotes the generation and dissemination of information, but also fuels the proliferation of disinformation. As one of the most popular types of content in disinformation, memes attract readers easily and brought further challenges to the detection of disinformation (Martino et al., 2020; Dimitrov et al., 2021).

Specifically, memes employ a number of techniques to influence users, which can be divided into the use of logical fallacies and appealing to the emotions of the audience (Dimitrov et al., 2021). In practice, the former misuses logical rules to disguise wrong conclusions as correct and objective,

\*Co-author.

†Corresponding author.

‡<https://www.163.com/dy/article/F0HKK63D0511EPAO.html>

§<http://www.zhujia120.com/fenxi/202103/318716.html>

¶<https://www.163.com/dy/article/G56M8U7R05521HYB.html>

	
GET US VACCINATED \n\n NOBODY CARES WHATS IN IT	PICK YOUR ROLE \n\n OR TAKE THE CHOICE MADE BY THEM
Name calling/Labeling Slogans Smears	Appeal to fear/prejudice Black-and-white Fallacy/Dictatorship

Figure 1: Examples of multi-modal samples, we rewrite the sentences on our own and collect the images from<sup>‡</sup>, <sup>§</sup>, <sup>¶</sup>, respectively. The first two rows illustrate the visual and the textual content, and in the last row, each line reveals the label (techniques) of the sample.

while the latter utilizes emotional language to induce the audience to agree with the speaker emotionally and prevent their rational analysis of the argumentation.

Identifying the techniques used in memes contributes to the understanding of user-generated content and further helps to the detection of disinformation. The subtask 3 of SemEval-2021 Task 6 (Dimitrov et al., 2021) is organized to stimulate the study of computational methods to detect persuasion techniques in memes that inhere in texts and images.

As shown in Figure 1, each sample consists of a set of textual sentences and an attached image. According to the task description (Dimitrov et al., 2021), the image and the sentence could convey the modality-specific persuasion techniques, respectively, and at the same time, images can be combined with sentence to express some techniques, which we named global techniques. Based on the understanding of the task, we attribute the main challenges of subtask 3 to the following three as-

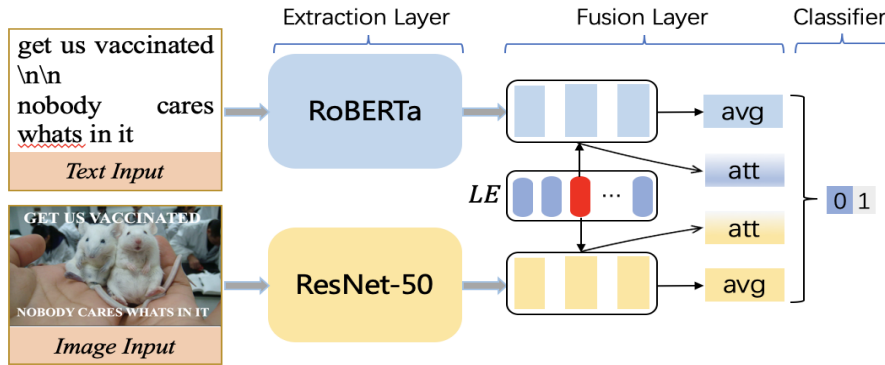


Figure 2: Overview of our proposed method. Our method takes the textual sentences and image as inputs and predicts a binary result for each label. Notably, “LE” in the figure denotes the label embedding module, and this figure illustrates the prediction for the label highlighted in red.

pects: 1) extracting essential features from each modality to predict modality-specific labels, 2) fusing multi-modal features to understand the content fully for predicting global labels, and 3) capturing the connections among multi labels.

Correspondingly, we present the methods to handle these challenges. Specifically, our method employs the powerful feature extractor including the pre-trained RoBERTa (Liu et al., 2019) and ResNet-50 (He et al., 2016) to extract textual and visual features, respectively. Besides, inspired by the work Augenstein et al. (2019), our method adopts a label embedding layer to learn how semantically close the labels are to one another implicitly, and the embedding layer maps each label to a learnable fixed-size vector. Before the label prediction, multi-modal features are fused according to their relevance with each label using attention mechanism (Bahdanau et al., 2015), and the final prediction is based on the fused features.

## 2 Related Work

Subtask 3 is a multi-label classification task based on multi-modal data. As for the multi-label classification tasks, it was earlier handled by many machine learning methods. Zhang et al. (Zhang and Zhou, 2005) used a k-nearest neighbor-based algorithm to conduct experiments on real-world multi-label bioinformatic data. Vens et al. (Vens et al., 2008) proposed a hierarchical multi-label classification method based on Decision trees. With the rapid development of deep learning, multi-label classification methods based on deep neural networks have become mainstream. Wang et al. (Wang et al., 2016) introduced and multi-label image classification network with the fusion of CNN and RNN.

Chernyavskiy et al. (Chernyavskiy et al., 2020) used a RoBERTa-based network combined with additional CRF (Lafferty et al., 2001) layers and transfer learning mechanism (Pan and Yang, 2010) to address a multi-label classification task in SemEval-2020. However, these previous multi-label classification tasks were often based on single modality data. These approaches fall short when the task requires the use of multiple modal data.

Moreover, in the field of multi-modal tasks, we focus on task of multi-modal fake news detection. Recent work (Jin et al., 2017; Wang et al., 2018; Khattar et al., 2019) mainly concern the fusion of multi-modal features and adopt a binary classifier, which is not applicable to current multi-label classification scenarios.

## 3 Methodology

### 3.1 Task Formulation

The task of identifying the techniques used in memes is defined as a multi-label classification problem of given multi-modal sample. We refer the textual sentences as  $\mathbf{S}$  and the attached image as  $\mathbf{I}$ , and use  $\mathcal{M}$  to denote the multi-modal model which map inputs  $\mathbf{S}$  and  $\mathbf{I}$  into a set of  $N$  binary values that represent the corresponding label. The task is formulated as follows:

$$\mathcal{M}(\mathcal{F}(\phi(\mathbf{S}), \phi(\mathbf{I}))) \longrightarrow \{0, 1, \dots, 1\} \quad (1)$$

In the Equation 1,  $\phi$  denotes the multi-modal feature extractor for textual and visual content, respectively, and  $\mathcal{F}$  denotes the fusion of the multi-modal features. The length of predicted results is the same as the number of labels and 1 indicates the corresponding label is predicted.

## 3.2 Method

In this section, we demonstrate the method used by our team for subtask 3. As shown in Figure 2, our method consists of three main layers: Extraction Layer, Fusion Layer, and the final Classifier. In the rest of this subsection, we describe each layer in detail.

### 3.2.1 Extraction Layer

In the Extraction Layer, the pre-trained RoBERTa is used to extract textual features. Specifically, given that RoBERTa receives at most two sentences as input while some samples may contain multiple pieces of sentences, we splice all sentences into a single sentence and retain the character “\n\n” as the separator. As for the outputs of RoBERTa, we merely reserve the representation of each token as sequential features  $\mathbf{T}$  for the post-processing.

For the image input, we use the ResNet-50 pre-trained on ImageNet to extract visual features. Before the image is input to the ResNet-50 network, it needs to be normalized and cropped into  $3*224*224$ . Afterward, we select the last convolution layer’s feature maps with size  $2048*7*7$  as visual features and transform it into a sequential features  $\mathbf{V}$  with size of  $49 * 2048$ .

### 3.2.2 Fusion Layer

The Fusion Layer aims to select the features for the label prediction. As mentioned earlier, the labels implied in the memes include both modality-specific labels and global labels. To promote the prediction of modality-specific labels, we perform the average-pooling on both textual and visual features to extract the modality-specific features  $\mathbf{T}_{avg}$  and  $\mathbf{V}_{avg}$  (“avg” in Figure 2).

$$\mathbf{T}_{avg} = \text{AvgPooling}(\mathbf{T}) \quad (2)$$

$$\mathbf{V}_{avg} = \text{AvgPooling}(\mathbf{V}) \quad (3)$$

Meanwhile, to promote global labels’ prediction, we adopt the attention mechanism to fuse multi-modal features. Particularly, As depicted in Equation 4-6, we first calculate the similarity between  $i$ th label embeddings and textual features. We then weighted-sum the textual features according to the similarity scores and obtain label-related representation  $\mathbf{T}_{i,att}$  (“att” in Figure 2). The similar operation is applied to the Visual and produce  $\mathbf{V}_{i,att}$ .

$$S_{i,j} = \mathbf{L}_i \cdot \mathbf{T}_j^T, \forall j \in [1, \dots, \ell_T] \quad (4)$$

$$\alpha_i = \text{Softmax}[S_{i,1}, \dots, S_{i,n}] \quad (5)$$

$$\mathbf{T}_{i,att} = \sum_{j=1}^{\ell_T} \alpha_{i,j} \mathbf{T}_j \quad (6)$$

Finally, we concatenate the features obtained above as the final representation of the input and pass it into the Classifier.

$$\mathbf{R}_i = [\mathbf{T}_{avg}; \mathbf{V}_{avg}; \mathbf{T}_{i,att}; \mathbf{V}_{i,att}] \quad (7)$$

### 3.2.3 Classifier

We adopt a three-layers fully connected network as the classifier, which maps the final representation  $\mathbf{R}_i$  obtained ahead into a scalar. Then we employ a sigmoid function to squeeze the scalar to the interval of 0-1. Notably, for each label, the process mentioned above is required and performed synchronously. Hence our model finally outputs a vector whose length is consistent with the number of labels.

## 4 Experimental Setup

### 4.1 Dataset

The dataset was provided by SemEval2021 Task6 subtask3, and the training set, development set, and test set contain 687, 63, and 200 samples, respectively. Each sample is combined with an image-text pair, id, and labels.

### 4.2 Evaluation Measures

The official evaluation measure for this technique classification is Micro-F1. The Macro-F1 is also reported, and we will consider both the performance of Micro-F1 and Macro-F1 during the experiment.

### 4.3 Parameter Settings

To train the model, we adopt the binary cross-entropy loss as the objective function and employ the Adam method(Kingma and Ba, 2015) with a learning rate of 0.0001 to optimize it. We set the minibatch size at 64 and the dimensions of label embeddings at 256. Based on experimental verification, we fixed the parameters of ResNet-50 while fine-tuning the parameters of RoBERTa during the training. Our methods are implemented with PyTorch and run on a single Nvidia 1080ti graphic card.

Model	Macro F1	Micro F1
RoBERTa+ ResNet-50	0.0812	0.1333
+ visual_att	0.1155	0.4140
+ textual_att	0.0814	0.5256
+ full_att	0.2040	0.5680

Table 1: Ablation results on validation set.

Rank	Team	Macro F1	Micro F1
1	Alpha	0.27315	0.58109
2	MinD	0.24389	0.56623
3	<b>1213Li</b>	<b>0.22830</b>	<b>0.54860</b>
4	AIMH	0.20729	0.53994
5	Volta	0.18877	0.52070
...	...	...	...
16	Baseline	0.05152	0.07062

Table 2: Evaluation results of top 5 teams on blind test set that reported on the official website.

#### 4.4 Ensemble

We use an ensemble of 5 models with different development set to predict the training set. Among the five ensembled models, one model uses the original training set and development set, and the remaining four models use the 64 samples randomly divided from the combined data of the training set and development set as the new development set, and use the rest as the training set.

### 5 Results and Discussion

The result of the ablation study is shown in Table 1. As we can see, the baseline method is very ineffective since it utilizes only the average-pooling features of visual and textual information, indicating that the lack of the interaction between modality-specific features and label information hinder the model to select vital features for prediction and leads to poor performance.

So we introduce the attention mechanism to selectively choose valid information from visual features and textual features, respectively. As shown in the second group of Table 1, the use of the attention mechanism significantly improves the model’s performance, especially the Micro F1 score.

Finally, the model that uses both visual features and textual features in combination with the attention mechanism has the optimal performance. During the test stage, we chose the model that performed best on the development set and got the

final result through the ensemble. The final evaluation results are reported in Table 2.

## 6 Conclusion

This paper demonstrates the method that we proposed for subtask 3 in SemEval-2021 Task 6, which aims to identify which of 22 persuasion techniques are used in the textual and visual content of the specific meme. Our method uses RoBERTa and ResNet-50 to extract multi-modal features, introduces the attention mechanism to fuse multi-modal features, and adopts the label embeddings to learn the representation of labels. Our proposed model achieves noticeable improvements over the baseline method, and the official evaluation ranked our submission 3rd out of 16 teams.

## References

- Isabelle Augenstein, Christina Lioma, Dongsheng Wang, Lucas Chaves Lima, Casper Hansen, Christian Hansen, and Jakob Grue Simonsen. 2019. [Multifc: A real-world multi-domain dataset for evidence-based fact checking of claims](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4684–4696. Association for Computational Linguistics.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. [Aschern at semeval-2020 task 11: It takes three to tango: Roberta, crf, and transfer learning](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation, SemEval@COLING 2020, Barcelona (online), December 12-13, 2020*, pages 1462–1468. International Committee for Computational Linguistics.
- Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Feroj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. [Task 6 at semeval-2021: Detection of persuasion techniques in texts and images](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval ’21, Bangkok, Thailand*.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. [Deep residual learning for image recognition](#). In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Ve-*

- gas, NV, USA, June 27-30, 2016, pages 770–778. IEEE Computer Society.
- Zhiwei Jin, Juan Cao, Han Guo, Yongdong Zhang, and Jiebo Luo. 2017. **Multimodal fusion with recurrent neural networks for rumor detection on microblogs**. In *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, pages 795–816. ACM.
- Dhruv Khattar, Jaipal Singh Goud, Manish Gupta, and Vasudeva Varma. 2019. **MVAE: multimodal variational autoencoder for fake news detection**. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 2915–2921. ACM.
- Diederik P. Kingma and Jimmy Ba. 2015. **Adam: A method for stochastic optimization**. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. **Conditional random fields: Probabilistic models for segmenting and labeling sequence data**. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **Roberta: A robustly optimized BERT pretraining approach**. *CoRR*, abs/1907.11692.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. **Semeval-2020 task 11: Detection of propaganda techniques in news articles**. *CoRR*, abs/2009.02696.
- Sinno Jialin Pan and Qiang Yang. 2010. **A survey on transfer learning**. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359.
- Celine Vens, Jan Struyf, Leander Schietgat, Saso Dzeroski, and Hendrik Blockeel. 2008. **Decision trees for hierarchical multi-label classification**. *Mach. Learn.*, 73(2):185–214.
- Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. 2016. **CNN-RNN: A unified framework for multi-label image classification**. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2285–2294. IEEE Computer Society.
- Yaqing Wang, Fenglong Ma, Zhiwei Jin, Ye Yuan, Guangxu Xun, Kishlay Jha, Lu Su, and Jing Gao. 2018. **EANN: event adversarial neural networks for multi-modal fake news detection**. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 849–857. ACM.
- Min-Ling Zhang and Zhi-Hua Zhou. 2005. **A k-nearest neighbor based algorithm for multi-label classification**. In *2005 IEEE International Conference on Granular Computing, Beijing, China, July 25-27, 2005*, pages 718–721. IEEE.



# NLyticsFKIE at SemEval-2021 Task 6: Detection of Persuasion Techniques In Texts And Images

**Albert Pritzkau**

Fraunhofer FKIE / Wachtberg, Germany  
albert.pritzkau@fkie.fraunhofer.de

## Abstract

The following system description presents our approach to the detection of persuasion techniques in texts and images. The given task has been framed as a multi-label classification problem with the different techniques serving as class labels. The multi-label classification problem is one in which a list of target variables such as our class labels is associated with every input chunk and assumes that a document can simultaneously and independently be assigned to multiple labels or classes.

In order to assign class labels to the given memes, we opted for RoBERTa (A Robustly Optimized BERT Pretraining Approach) as a neural network architecture for token and sequence classification. Starting off with a pre-trained model for language representation we fine-tuned this model on the given classification task with the provided annotated data in supervised training steps. To incorporate image features in the multi-modal setting, we rely on the pre-trained VGG-16 model architecture.

## 1 Introduction

Social networks provide opportunities to conduct disinformation campaigns for organizations as well as individual actors. The proliferation of disinformation online, has given rise to a lot of research on automatic fake news detection. SemEval-2021 Task 6 considers disinformation as a communication phenomenon. By detecting the use of various persuasion techniques in communication, it takes into account not only the content but also how a subject matter is communicated.

The goal of the shared task is to build models for identifying such techniques in the textual content of a meme only (two subtasks) and in a multimodal setting in which both the textual and the visual content are to be analysed together (one subtask).

The shared task defines the following subtasks:

### Subtask 1

Given only the “textual content” of a meme, identify which of the 20 techniques are used in it. This is a multi-label classification problem.

### Subtask 2

Given only the “textual content” of a meme, identify which of the 20 techniques are used in it together with the span(s) of text covered by each technique. This is a multilabel sequence tagging task. The task is the combination of the two subtasks of the SemEval 2020 task 11 on “detecting propaganda techniques in news articles”. Note that subtask 1 is a simplified version of subtask 2 in which the spans covered by each technique is not supposed to be provided.

### Subtask 3

Given a meme, identify which of the 22 techniques are used both in the textual and visual content of the meme (multi-modal task). This is a multilabel classification problem.

In this work, we covered our approach on both technique classification (TC) tasks (Subtask 1 and Subtask 3) detecting the type of communication technique used in a given message. To build models, the first subtask assumes purely textual content as inputs, whereas the third is designed in multi-modal setting in which both the textual and the visual content are to be analysed together. Below, we describe the systems built for these two subtasks. At the core of our systems is RoBERTa (Liu et al., 2019), a pre-trained model based on the Transformer architecture (Vaswani et al., 2017).

Although we did not manage to participate in

the second subtask, we will describe our solution below for the sake of completeness.

Finally, we will address some limitation of the general settings of this shared task.

## 2 Related Work

The goal of the shared task is to investigate automatic techniques for identifying various rhetorical and psychological techniques in online disinformation campaigns. A comprehensive survey on fake news has been presented by [Zhou and Zafarani \(2018\)](#). Based on the structure of data reflecting different aspects of communication, they identified four different perspectives on fake news: (1) the false knowledge it carries, (2) its writing style, (3) its propagation patterns, and (4) the credibility of its creators and spreaders.

The shared task emphasizes communicative styles that systematically co-occur with persuasive intentions of (political) media actors. Similar to [de Vreese et al. \(2018\)](#), propaganda and persuasion is considered as an expression of political communication content and style. Hence, beyond the actual subject of communication, the way it is communicated is gaining importance.

We build our work on top of this foundation by first investigating content-based approaches for information discovery and then open up our focus to dissemination mechanisms. Traditional information discovery methods are based on content: documents, terms, and the relationships between them ([Leskovec and Lang, 2008](#)). They can be considered as a general Information Extraction (IE) methods, automatically deriving structured information from unstructured and/or semi-structured machine-readable documents. Communities of researchers contributed various techniques from machine learning, information retrieval, and computational linguistics to the different aspects of the information extraction problem. From a computer science perspective, existing approaches can be roughly divided into the following categories: rule-based, supervised, and semi-supervised. In our case, we followed the supervised approach by reframing the complex language understanding task as a simple classification problem. Text classification also known as text tagging or text categorization is the process of categorizing text into organized groups. By using Natural Language Processing (NLP), text classifiers can automatically analyze human language texts and then assign a set

of predefined tags or categories based on their content. Historically, the evolution of text classifiers can be divided into three stages: (1) simple lexicon- or keyword-based classifiers, (2) classifiers using distributed semantics, and (3) deep learning classifiers with advanced linguistic features.

### 2.1 Deep Learning for IE

Recent work on text classification uses neural networks, particularly Deep Learning (DL). [Badjatiya et al. \(2017\)](#) demonstrated that these architectures, including variants of recurrent neural networks (RNN) ([Gao and Huang, 2017](#); [Pavlopoulos et al., 2017](#); [Pitsilis et al., 2018](#)), convolutional neural networks (CNN) [Zhang et al. \(2018\)](#), or their combination (CharCNN, WordCNN, and HybridCNN), produce state-of-the-art results and outperform baseline methods (character n-grams, TF-IDF or bag-of-words representations).

### 2.2 Deep Learning architectures

Until recently, the dominant paradigm in approaching NLP tasks has been focused on the design of neural architectures, using only task-specific data and word embeddings such as those mentioned above. This led to the development of models, such as Long Short Term Memory (LSTM) networks or Convolution Neural Networks (CNN), that achieve significantly better results in a range of NLP tasks than less complex classifiers, such as Support Vector Machines, Logistic Regression or Decision Tree Models. [Badjatiya et al. \(2017\)](#) demonstrated that these approaches outperform models based on char and word n-gram representations. In the same paradigm of pre-trained models, methods like BERT ([Devlin et al., 2018](#)) and XLNet ([Yang et al., 2019](#)) have been shown to achieve the state of the art in a variety of tasks.

### 2.3 Pre-trained Deep Language Representation Model

Indeed, the usage of a pre-trained word embedding layer to map the text into vector space which is then passed through a neural network, marked a significant step forward in text classification. The potential of pre-trained language models, as e.g. Word2Vec ([Mikolov et al., 2013](#)), GloVe ([Pennington et al., 2014](#)), fastText ([Joulin et al., 2017](#)), or ELMo ([Peters et al., 2018](#)) to capture the local patterns of features to benefit text classification, has been described by [Castelle \(2019\)](#). Modern pre-trained language models use unsupervised learning

techniques such as creating RNNs embeddings on large texts corpora to gain some primal “knowledge” of the language structures before a more specific supervised training steps in.

## 2.4 About BERT and RoBERTa

BERT stands for Bidirectional Encoder Representations from Transformers. It is based on the Transformer model architectures introduced by Vaswani et al. (2017). The general approach consists of two stages: first, BERT is pre-trained on vast amounts of text, with an unsupervised objective of masked language modeling and next-sentence prediction. Second, this pre-trained network is then fine-tuned on task specific, labeled data. The Transformer architecture is composed of two parts, an Encoder and a Decoder, for each of the two stages. The Encoder used in BERT is an attention-based architecture for NLP. It works by performing a small, constant number of steps. In each step, it applies an attention mechanism to understand relationships between all words in a sentence, regardless of their respective position. By pre-training language representations, the Encoder yields models that can either be used to extract high quality language features from text data, or fine-tune these models on specific NLP tasks (classification, entity recognition, question answering, etc.). We rely on RoBERTa (Liu et al., 2019), a pre-trained Encoder model which builds on BERT’s language masking strategy. However, it modifies key hyperparameters in BERT such as removing BERT’s next-sentence pre-training objective, and training with much larger mini-batches and learning rates. Furthermore, RoBERTa was also trained on an order of magnitude more data than BERT, for a longer amount of time. This allows RoBERTa representations to generalize even better to downstream tasks compared to BERT. In this study, RoBERTa is at the core of each solution of the given subtasks.

## 2.5 Image Feature Extraction using Pre-trained Models

Convolutional neural network (CNN) visual features have demonstrated their powerful ability as a universal representation for various recognition tasks. In this study we rely on the extraction of visual features on state of the art convolutional neural network architectures. From the most popular architectures such as VGG (Simonyan and Zisserman, 2015), ResNet (He et al., 2016), AlexNet (Krizhevsky et al., 2017), GoogLeNet (Szegedy

et al., 2015) we initially generated the image features using a pre-trained VGG-16 model architecture.

## 2.6 Multimodal Deep Learning

Multimodal deep learning involves multiple modalities used together to predict some output. The different modalities present in a particular piece of content are extracted and fused early in the classification process. In this study, we concatenated the features extracted from images and text sequences using a Convolutional Neural Network (CNN) and RoBERTa encodings (Liu et al., 2019), respectively. These features were used to try and predict persuasive techniques.

## 3 Dataset

The dataset to this task is provided by Dimitrov et al. (2021). Furthermore, there is a related shared task “SemEval 2020 task 11 on Detecting propaganda techniques in news articles” (Martino et al., 2020) since it serves as the basis for the second sub-task. In particular, the second subtask is the combination of the two subtasks of the previous task. Finally, there is a recent survey on computational propaganda detection by da San Martino et al. (2019).

## 4 Our approach

In this section, we provide a general overview of our approaches to the three subtasks. Subtasks 1 and 3 are both given as multilabel classification problems, whereas subtasks 2 is given as a multilabel sequence tagging task.

### 4.1 Experimental setup: Subtask 1

**Model Architecture** This subtask is a multi-class multi-label problem, as one or more labels have to be assigned to each sample. Our model for this subtask is based on RoBERTa.

**Input Embeddings** The input embedding layer converts the inputs (memes text) into sequences of features: word-level sentence embeddings. These embedding features will be further processed by the latter encoding layers.

**Word-Level Sentence Embeddings** A sentence is split into words  $w_1, \dots, w_n$  with length of  $n$  by the WordPiece tokenizer (Wu et al., 2016). The word  $w_i$  and its index  $i$  ( $w_i$ ’s absolute position in the sentence) are projected to vectors by embedding

sub-layers, and then added to the index-aware word embeddings:

$$\hat{w}_i = \text{WordEmbed}(w_i)$$

$$\hat{u}_i = \text{IdxEmbed}(i)$$

$$h_i = \text{LayerNorm}(\hat{w}_i + \hat{u}_i)$$

**Target Encoding** We encode the target labels using a multi-label binarizer as an analog of one-hot aka one-of-K scheme to multiple labels.

## 4.2 Experimental setup: Subtask 2

This subtask is given as a multilabel sequence tagging problem.

**Tagging format** We transformed the initial span markup into a IOB tagging format (Inside, Outside, Begin). As we have 20 possible entity classes, each token can be assigned one of the 41 tags given by an O-tag, and the I-tag and B-tag of the various techniques, respectively.

**Model Architecture** We fine-tuned a RoBERTa model to predict the above IOB tags for each token in the input sentence. One problem with the above setup is that each token is classified independently of the surrounding tokens: while these surrounding tokens are taken into account in the contextualized embeddings that RoBERTa produces, there is no modeling of the dependency between the predicted labels: for example, logically an I-tag should not follow O. Since RoBERTa does not model the dependencies between the predicted token, we further added a linear-chain Conditional Random Field (CRF) model (Lafferty et al., 2001) as an additional layer, in order to model the dependency between the predicted labels of individual tokens. Since the sequence of an O-tag following an I-tag does not appear in the training set, it assigns by observation a very low probability to the transition from an O-tag to an I-tag. We trained the resulting RoBERTa-CRF model as shown in Figure 1. The CRF receives the logits for each input token, and makes a prediction for the entire input sequence, taking into account the dependencies between the labels, similarly to (Lample et al., 2016). Note that RoBERTa works with byte pair encoding (BPE) units, while for the CRF it makes more sense to work with complete words. Thus, only head tokens were used as input to the CRF, and skipping any word continuation tokens.

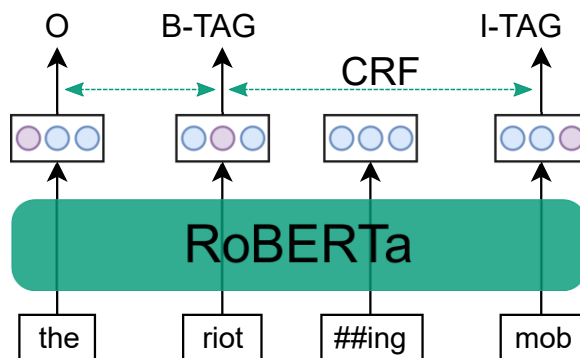


Figure 1: RoBERTa-CRF model with IOB-encoded target. The CRF model ignores non-starting word pieces such as the depicted ##ing token.

## 4.3 Experimental setup: Subtask 3

**Model Architecture** We build our cross-modality model with self-attention and cross-attention layers following the recent progress in designing natural language processing models (e.g., transformers (Vaswani et al., 2017)). Our model takes two inputs as part of a meme: an image and its related text. Each image is represented as a feature vector, and each sentence is represented as a sequence of words. As depicted in Figure 2, via design and combination of the self-attention and cross-attention layers, our model is able to generate language representations, image representations, and cross-modality representations from the inputs. Next, we describe the components of this model in detail.

**Input Embeddings** The input embedding layers convert the inputs (i.e., an image and a short text) into two sequences of features: word-level sentence embeddings and image embeddings. These embedding features will be further processed by the latter encoding layers.

**Word-Level Sentence Embeddings** A sentence is split into words  $w_1, \dots, w_n$  with length of  $n$  by the WordPiece tokenizer (Wu et al., 2016). The word  $w_i$  and its index  $i$  ( $w_i$ 's absolute position in the sentence) are projected to vectors by embedding sub-layers, and then added to the index-aware word embeddings:

$$\hat{w}_i = \text{WordEmbed}(w_i)$$

$$\hat{u}_i = \text{IdxEmbed}(i)$$

$$h_i = \text{LayerNorm}(\hat{w}_i + \hat{u}_i)$$

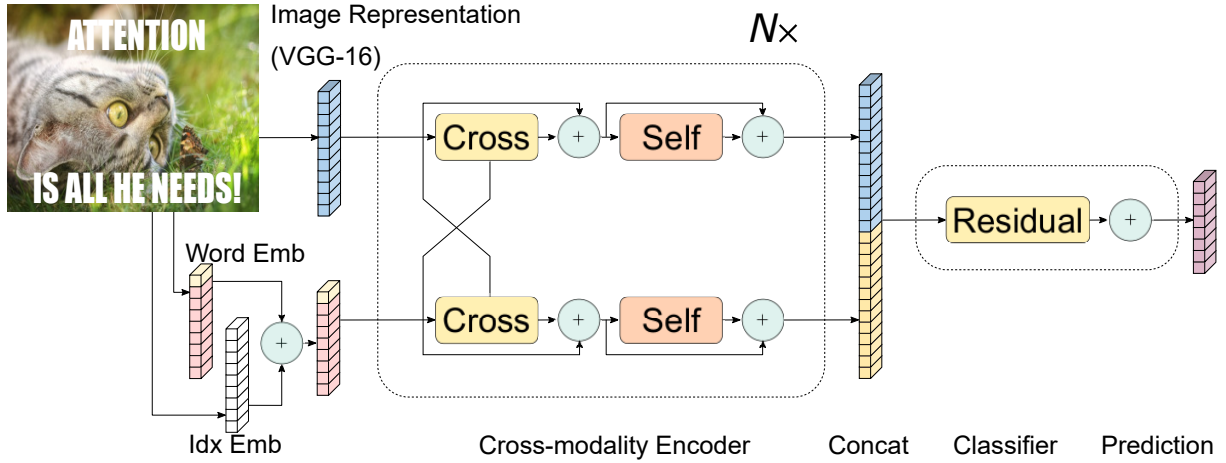


Figure 2: Multimodal model configuration for learning image-and-text cross-modality representations. ‘Self’ and ‘Cross’ are abbreviations for self-attention sublayers and cross-attention sublayers, respectively.

**Visual features** In this study we rely on the extraction of visual features on state of the art convolutional neural network architectures by generating the image features using a pre-trained VGG-16 model.

**Encoders** We build our encoders, i.e., the language encoder, and the cross-modality encoder, on the basis of two kinds of attention layers: self-attention layers and crossattention layers. We first review the definition and notations of attention layers and then discuss how they form our encoders.

**Attention Layers** Attention layers (Bahdanau et al., 2015; Xu et al., 2015) aim to retrieve information from a set of context vectors  $y_j$  related to a query vector  $x$ . An attention layer first calculates the matching score  $a_j$  between the query vector  $x$  and each context vector  $y_j$ . Scores are then normalized by softmax:

$$a_j = \text{score}(x, y_j)$$

$$\alpha_j = \exp(a_j) / \sum_k \exp(a_k)$$

The output of an attention layer is the weighted sum of the context vectors w.r.t. the softmax normalized score:  $Att_{X \rightarrow Y}(x, \{y_j\}) = \sum_j \alpha_j y_j$ . An attention layer is called self-attention when the query vector  $x$  is in the set of context vectors  $y_j$ . Specifically, we use the multi-head attention following Transformer (Vaswani et al., 2017).

**Single-Modality Encoders** After the embedding layers, we apply a temporal convolutional layer to each single modality. The result of this projection is a uniform feature space with defined

dimensions as the input to the cross-modality encoder.

**Cross-Modality Encoder** Each cross-modality layer in the cross-modality encoder consists of two self-attention sub-layers, one bi-directional cross-attention sub-layer, and a feed-forward sub-layer. We stack (i.e., using the output of  $k$ -th layer as the input of  $(k+1)$ -th layer)  $N \times$  these cross-modality layers in our encoder implementation. Inside the  $k$ -th layer, the bi-directional cross-attention sub-layer (‘Cross’) is first applied, which contains two unidirectional cross-attention sub-layers: one from text to image and one from image to text. The query and context vectors are the outputs of the  $(k-1)$ -th layer (i.e., text features  $\{t_i^{k-1}\}$  and image features  $\{i_j^{k-1}\}$ ):

$$\hat{t}_i^k = \text{CrossAtt}_{T \rightarrow I}(t_i^{k-1}, \{i_1^{k-1}, \dots, i_m^{k-1}\})$$

$$\hat{i}_j^k = \text{CrossAtt}_{I \rightarrow T}(i_j^{k-1}, \{t_1^{k-1}, \dots, t_n^{k-1}\})$$

The cross-attention sub-layer is used to exchange the information and align the entities between the two modalities in order to learn joint cross-modality representations. For further building internal connections, the self-attention sub-layers (‘Self’) are then applied to the output of the crossattention sub-layer:

$$\tilde{t}_i^k = \text{SelfAtt}_{T \rightarrow T}(\hat{t}_i^k, \{\hat{t}_1^k, \dots, \hat{t}_m^k\})$$

$$\tilde{i}_j^k = \text{SelfAtt}_{I \rightarrow I}(\hat{i}_j^k, \{\hat{i}_1^k, \dots, \hat{i}_n^k\})$$

We add a residual connection and layer normalization (annotated by the ‘+’ sign in Fig. 1) after each sublayer as in Vaswani et al. (2017).

**Classification** At the core of the classifier we use a residual block as introduced by He et al. (2016). The input to the residual block is given by concatenation of the output of the cross-modality encoder. The input and output size of the residual block corresponds to the sum of the output size of the cross-modality encoder. Lastly, in order to obtain the desired one-hot encoding as the output of the classifier, a linear transformation is applied.

**Target Encoding** We encode the target labels using a multi-label binarizer as an analog of one-hot aka one-of-K scheme to multiple labels.

#### 4.4 Results and Discussion

We participated in both techniques classification tasks (subtask 1 and 3). The official evaluation ranked our system 9th and 13th out of 16 and 15 teams, respectively. In this study, we focused on suitable combinations deep learning methods as well as their hyperparameter settings. Finetuning pre-trained language models like RoBERTa on downstream tasks has become ubiquitous in NLP research and applied NLP. Even without extensive pre-processing of the training data, we already achieve competitive results and can serve as strong baseline models which, when fine-tuned, significantly outperform training models from scratch. When improving on these baseline models, data scarcity appears to be an immense challenge. This is especially evident in the ratio of the given training samples to the number of possible target classes. We expected better results with the multimodal solution. The causes of the problem will be investigated in more detail in the future.

## 5 Conclusion and Future work

We described our approach for the SemEval-2021 Task 6 on Detection of Persuasion Techniques in Text and Images. We employed RoBERTa-based neural architectures, additional CRF layers, and a cross-modality framework for learning the connections between image and text in a multi-modal transformer architecture.

In future work, we plan to investigate more recent neural architectures for language representation such as T5 (Raffel et al., 2019) and GPT-3 (Brown et al., 2020). In case of the multimodal setting, it might also be useful to evaluate alternative model architectures such as ResNet (He et al., 2016) to improve image representation.

Furthermore, we expect great opportunities for transfer learning from the areas such as argumentation mining (Stede, 2020) and offensive language detection (Zampieri et al., 2019). To deal with data scarcity as a general challenge in natural language processing, we examine the application of concepts such as active learning, semi-supervised learning (Ruder and Plank, 2018) as well as weak supervision (Ratner et al., 2020).

## References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. [Deep learning for hate speech detection in tweets](#). In *26th International World Wide Web Conference 2017, WWW 2017 Companion*, pages 759–760. International World Wide Web Conferences Steering Committee.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. [Neural machine translation by jointly learning to align and translate](#). In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. [Language models are few-shot learners](#).
- Michael Castelle. 2019. [The Linguistic Ideologies of Deep Abusive Language Classification](#). pages 160–170.
- Giovanni da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. [Fine-grained analysis of propaganda in news articles](#).
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#).
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Feroj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at SemEval-2021: Detection of Persuasion Techniques in Texts and Images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval’21*, Bangkok, Thailand.

- Lei Gao and Ruihong Huang. 2017. [Detecting online hate speech using context aware models](#). In *International Conference Recent Advances in Natural Language Processing, RANLP*, volume 2017-Sept, pages 260–266. Association for Computational Linguistics (ACL).
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. [Deep residual learning for image recognition](#). In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 770–778. IEEE Computer Society.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. [Bag of tricks for efficient text classification](#). In *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, volume 2, pages 427–431.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2017. [ImageNet classification with deep convolutional neural networks](#). *Communications of the ACM*, 60(6):84–90.
- John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. [Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data](#). *Abstract*. 2001(June):282–289.
- Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. [Neural architectures for named entity recognition](#). In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*, pages 260–270. Association for Computational Linguistics (ACL).
- J Leskovec and KJ Lang. 2008. [Statistical properties of community structure in large social and information networks](#). *Proceedings of the 17th international conference on World Wide Web. ACM*, pages 695–704.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#).
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. [SemEval-2020 Task 11: Detection of propaganda techniques in news articles](#).
- Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. [Exploiting Similarities among Languages for Machine Translation](#).
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deeper attention to abusive user content moderation](#). In *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, pages 1125–1135, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. [GloVe: Global vectors for word representation](#). In *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1532–1543.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep Contextualized Word Representations](#). pages 2227–2237. Association for Computational Linguistics (ACL).
- Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. [Effective hate-speech detection in Twitter data using recurrent neural networks](#). *Applied Intelligence*, 48(12):4730–4742.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. [Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer](#). *arXiv*, 21:1–67.
- Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2020. [Snorkel: rapid training data creation with weak supervision](#). In *VLDB Journal*, volume 29, pages 709–730. Springer.
- Sebastian Ruder and Barbara Plank. 2018. [Strong Baselines for Neural Semi-supervised Learning under Domain Shift](#). *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 1:1044–1054.
- Karen Simonyan and Andrew Zisserman. 2015. [Very deep convolutional networks for large-scale image recognition](#). In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.
- Manfred Stede. 2020. [Automatic argumentation mining and the role of stance and sentiment](#). *Journal of Argumentation in Context*, 9(1):19–41.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. [Going deeper with convolutions](#). In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 1–9.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 5999–6009.
- Claes H. de Vreese, Frank Esser, Toril Aalberg, Carsten Reinemann, and James Staney. 2018. [Populism as an Expression of Political Communication Content and Style: A New Perspective](#). *International Journal of Press/Politics*, 23(4):423–438.

- Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. [Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation](#).
- Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. [Show, attend and tell: Neural image caption generation with visual attention](#). In *32nd International Conference on Machine Learning, ICML 2015*, volume 3, pages 2048–2057. International Machine Learning Society (IMLS).
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [XLNet: Generalized Autoregressive Pretraining for Language Understanding](#). Technical report.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [Predicting the type and target of offensive posts in social media](#). In *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, volume 1, pages 1415–1420, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. [Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network](#). In *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10843 LNCS, pages 745–760. Springer Verlag.
- Xinyi Zhou and Reza Zafarani. 2018. [Fake News: A Survey of Research, Detection Methods, and Opportunities](#). *ACM Comput. Surv.*, 1(1).



# YNU-HPCC at SemEval-2021 Task 6: Combining ALBERT and Text-CNN for Persuasion Detection in Texts and Images

Xingyu Zhu, Jin Wang and Xuejie Zhang  
School of Information Science and Engineering  
Yunnan University  
Kunming, China

Contact: xyzhu@mail.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

## Abstract

In recent years, memes combining image and text have been widely used in social media, and memes are one of the most popular types of content used in online disinformation campaigns. In this paper, our study on the detection of persuasion techniques in texts and images in SemEval-2021 Task 6 is summarized. For propaganda technology detection in text, we propose a combination model of both ALBERT and Text-CNN for text classification, as well as a BERT-based multi-task sequence labeling model for propaganda technology coverage span detection. For the meme classification task involved in text understanding and visual feature extraction, we designed a parallel channel model divided into text and image channels. Our method<sup>1</sup> achieved a good performance on subtasks 1 and 3. The micro  $F_1$ -scores of 0.492, 0.091, and 0.446 achieved on the test sets of the three subtasks ranked 12th, 7th, and 11th, respectively, and all are higher than the baseline model.

## 1 Introduction

The intentional shaping of information to promote a predetermined agenda is called propaganda. Propaganda uses psychological and rhetorical techniques to achieve its purpose. Propaganda techniques generally include the use of logical fallacies and appeal to the emotions of the audience. In recent years, memes combining images and text have been widely used in social media, and the use of memes can easily and effectively attract a large number of users on social platforms. Memes are one of the most popular types of content used in online disinformation campaigns (Martino et al., 2020), and memes applied in a disinformation campaign achieve their purpose of influencing users through

rhetorical and psychological techniques. Therefore, it is meaningful to research computational techniques for automatically detecting propaganda in particular content.

The SemEval 2021 Task 6 (Dimitrov et al., 2021) consists of three subtasks:

- Subtask 1 - Given only the “textual content” of a meme, identify which of the 20 techniques are used. The 20 techniques include appeal to authority, loaded language, and name calling or labeling.
- Subtask 2: Given only the “textual content” of a meme, identify which of the 20 techniques are used along with the span(s) of the text covered by each technique.
- Subtask 3: Given a meme, identify which of the 22 techniques are used for both the textual and visual content of the meme. These 22 technologies include the 20 technologies in subtasks 1 and 2, and 2 technologies, i.e., transfer and appeal to (strong) emotions, are added.

The detection of propaganda techniques in texts is similar to a text sentiment analysis, and both can be attributed to text classification tasks. In a previous study, Peng et al. (2020) used the adversarial learning of sentiment word representations for a sentiment analysis. A tree-structured regional CNN-LSTM (Wang et al., 2020) and dynamic routing in a tree-structured LSTM (Wang et al., 2019) were used for a dimensional sentiment analysis. In previous SemEval competitions, Dao et al. (2020) used GloVe-LSTM and BERT-LSTM models, and Paraschiv et al. (2020) used an ensemble model containing BERT and BiLSTM to detect both spans and categories of propaganda techniques in news articles (Da San Martino et al., 2020). In addition, in multimodal analysis combining images and

<sup>1</sup>The code of this paper is available at: <https://github.com/zxyqjuring/SemEval-2021-task6>

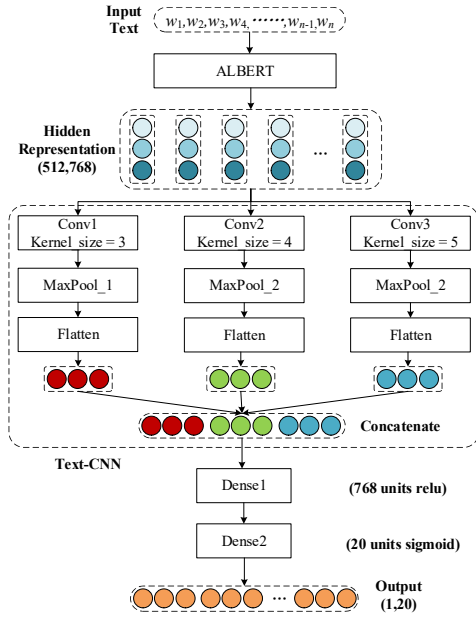


Figure 1: ALBERT-Text-CNN model architecture.

text, Yuan et al. (2020) proposed a parallel channel ensemble model combining BERT embedding, BiLSTM, attention and CNN, and ResNet for a sentiment analysis of memes. Li et al. (2019) proposed a Visual BERT model that aligns and fuses text and image information using transformers (Vaswani et al., 2017).

In this paper, we propose three different systems for the three subtasks in SemEval-2021 Task 6. For subtask 1, we added a Text-CNN layer after the pre-trained model ALBERT to fine-tune it for a multi-label classification of text. For subtask 2, we used the idea of partitioning to transform the problem into the detection of 20 techniques for each text separately. BERT was used in the model for text feature extraction followed by multi-task sequence labeling, and the results of each task were combined to obtain the final results. For subtask 3, we built the system using a parallel channel model containing text and image channels. The text channel used both the ALBERT and Text-CNN models to extract features of text in the meme, and the image channel used ResNet and VGGNet for image feature extraction. The information extracted by the two parallel channels was then combined through a fully connected layer after concatenation. Using micro  $F_1$ -scores as metrics, the results of the proposed model in subtasks 1, 2, and 3 were 0.625, 0.215, and 0.636, respectively, on the dev set.

The remainder of this paper is organized as follows. First, section 2 describes the details of the

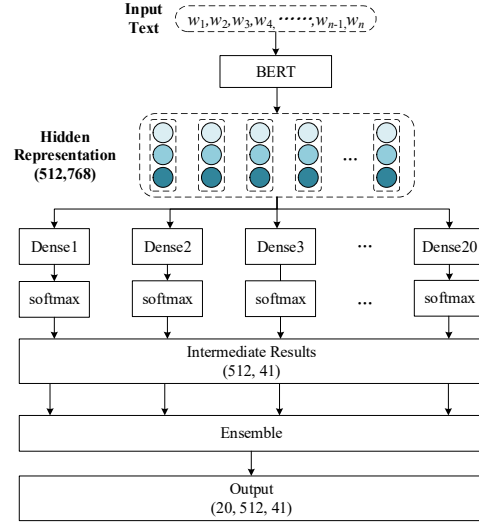


Figure 2: Architecture of multi-task sequence labeling model.

ALBERT and Text-CNN used in our system. Section 3 then presents the experimental results. Finally, some concluding remarks are presented in section 4.

## 2 System Overview

### 2.1 Subtask 1

Subtask 1 requires a detection model that uses only the textual features of the meme content and detects which of the 20 propaganda techniques were used. This is a multi-label classification problem for text, based on the pre-trained ALBERT model and added a Text-CNN layer. As illustrated in Figure 2, the proposed model includes an ALBERT layer, a Text-CNN layer, a fully connected layer, and an output layer.

- ALBERT (Lan et al., 2020) is a lite BERT for self-supervised learning of language representations, which uses layer-to-layer parameter sharing to reduce the number of parameters of the model, which not only speeds up the model training but also outperforms BERT on certain datasets. With our model, the pre-trained ALBERT model is fine-tuned to obtain a  $512 \times 768$  hidden representation matrix for subsequent multi-label classification of text.
- Text-CNN (Kim, 2014) is a convolutional neural network applied to a text classification task, using multiple kernels of different sizes to extract key information in sentences, and is thus able to better capture the local relevance. In

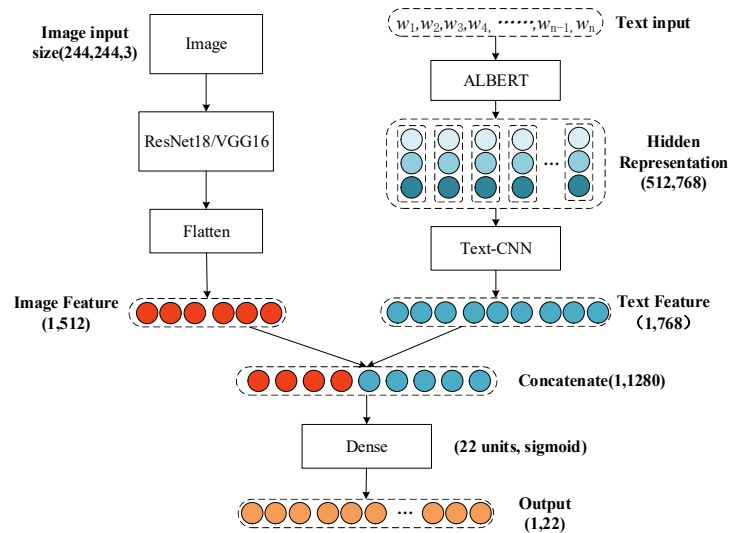


Figure 3: Architecture of parallel channel model.

this layer, we used three different sizes of one-dimensional convolution kernels, i.e., 3, 4, and 5, to extract information from the hidden representation matrix output from the ALBERT layer for the final multi-label text classification.

## 2.2 Subtask 2

Subtask 2 was a multi-label sequence-labeling task. We built the model by converting the problem to detect the coverage of each propagation technique separately for the input sequence, and built a multi-task sequence labeling model based on a fine-tuning of BERT.

As illustrated in Figure 3, the input sequence was first obtained using the pre-trained BERT (Devlin et al., 2019) model with a hidden representation matrix with dimensions of  $512 \times 768$ . Subsequently, 20 parallel fully connected layers were input separately for the detection of each propaganda technique coverage span (For each propagation technique, the sequence labeling task is performed separately for the input text). For each technique, the intermediate result of each parallel channel output is a  $512 \times 41$  matrix, and the ensemble layer represents the stacking of 20 matrices from 20 parallel channels, the dimensions of the final output were  $20 \times 512 \times 41$ , which denote the propaganda technique category, maximum sentence length, and code corresponding to each technique, respectively.

## 2.3 Subtask 3

For subtask 3, we modeled the problem as a multi-label classification task of the meme text and image content. We used a parallel channel model of text and image channels, and then concatenated the text and image features extracted by the two parallel channels to apply multi-label meme classification. The architecture of the proposed model is shown in Figure 4.

**Text Channel.** In the text channel, we used the ALBERT-Text-CNN model used in subtask 1, taking the text part of the meme content as an input to obtain a 768-dimensional text feature vector as the output.

**Image Channel.** In the image channel, we used ResNet and VGGNet, taking the image part of the meme content as input to obtain a 512-dimensional image feature vector as the output. The ResNet model (He et al., 2016) is a deep residual learning model for image recognition, and presents the inter-layer residual jump connection and solves the deep vanishing gradient problem. VGGNet (Simonyan and Zisserman, 2015) is a deep convolutional neural network with small-sized convolutional kernels and a regular network structure, in which the size of the convolution kernels used in VGG16 in our experiment is  $3 \times 3$ , and the pooling kernels is  $2 \times 2$ . Furthermore, only the structures of the ResNet and VGGNet were used in our experiment, and the pre-training weights were not applied.

### 3 Experimental Results

#### 3.1 Dataset

The organizer provided a dataset containing 687 memes for the training set, 63 memes for the development set, and 200 memes for the test set. The dataset of subtask 1 provides the ID, text of the meme, and the corresponding propaganda techniques used, and the dataset of subtask 3 also contains the corresponding meme image. The dataset of subtask 2 provides the ID, text of the meme, and the corresponding propaganda techniques used in a certain text fragment, in which the scope covered by the propaganda technology in the text is marked as “start,” “end,” and “text fragment,” respectively.

The datasets were preprocessed using the following procedures before model training:

- In subtasks 1 and 3, we first used one-hot encoding to encode the label into a vector whose length is the total number of technology categories.
- In subtask 2, we labeled each token in the text as “I-technique” and “O-technique” based on the 20 propaganda technology terms. “I-technique” indicates that the publicity technique was used and “O-technique” indicates that it was not, e.g., O-Smears and I-Smears. For 20 different propaganda techniques there are 40 different codes, and then add another padding code, so the label code length is 41.
- In subtask 3, we normalized the meme image size to  $224 \times 224 \times 3$ .

#### 3.2 Evaluation Metrics

The official evaluation measure for all subtasks is the micro  $F_1$ -score, which is defined as follows:

$$F_1 - score = 2 * \frac{Prec * Rec}{Prec + Rec} \quad (1)$$

where  $Prec$  and  $Rec$  denote the precision and recall scores of all samples, respectively. For subtask 2, the standard micro  $F_1$ -score was slightly modified to account for partial matching between spans (Dimitrov et al., 2021). In addition, the macro  $F_1$ -score was also reported for each type of propaganda.

#### 3.3 Implementation Details

All models used the TensorFlow2 backend, and all BERT-based models were implemented using

the HuggingFace Transformers toolkit(Wolf et al., 2020). The Adam optimizer (Ba and Kingma, 2015) was used to update all trainable parameters. The loss functions in subtasks 1 and 3 were binary cross-entropy, and subtask 2 was categorical cross-entropy. The hyper-parameters in the model training process were obtained using a grid-search strategy, as shown in Table 1. Once the optimal settings of the parameters were obtained, they were used for classification on the test sets of different corpora.

Hyper-parameter	Values
Learning rate	5e-6
Adam epsilon	1e-8
Text-CNN dropout	0.3
Text-CNN filters	64
Classifier dropout	0.3
Batch size	16

Table 1: Hyper-parameters in our models.

#### 3.4 Results

Table 2 presents the results of Subtask 1. We conducted experiments on several pre-trained models including BERT, RoBERTa(Liu et al., 2019), and ALBERT combined with the Text-CNN layer, and observed that the ALBERT and Text-CNN models achieved the best performance, the reason for which may be that the training datasets are small, and a serious overfitting will occur by directly fine-tuning BERT. Furthermore, the experiments show that the ALBERT model has fewer parameters and performs better on small datasets. Adding a Text-CNN layer after the BERT-based model can better extract the local relevance information of the text, which not only effectively alleviates the overfitting phenomenon it also effectively improves the model performance.

In subtask 2, the results of our proposed multi-task sequence labeling model on the dev set are  $F_1$ -score of 0.215,  $Precision$  of 0.378, and  $Recall$  of 0.151. The results on the test set are  $F_1$ -score of 0.091,  $Precision$  of 0.186, and  $Recall$  of 0.061.

Table 3 shows the results of Subtask 3. It can be observed that ResNet18 works better than VGG16 when using both ALBERT and ALBERT-Text-CNN models. The performance was improved by adding a Text-CNN layer to the text channel. Considering that the micro  $F_1$ -scores are relatively close, we selected the models with the top-three  $F_1$ -

Model	$F_1$ -Macro	$F_1$ -Micro
BERT	0.302	0.509
RoBERTa-Text-CNN	0.385	0.500
BERT-Text-CNN	0.414	0.560
ALBERT-Text-CNN	<b>0.472</b>	<b>0.625</b>

Table 2: Scores of different models for subtask 1 on dev set.

Model	$F_1$ -Macro	$F_1$ -Micro
ALBERT-VGG16	0.240	0.577
ALBERT-ResNet18	0.272	0.605
ALBERT-Text-CNN+VGG16	<b>0.346</b>	0.606
ALBERT-Text-CNN+ResNet18	0.247	<b>0.636</b>
Hard Voting	0.245	0.625

Table 3: Experimental results of different models for subtask 3 on dev set.

scores and used hard voting to generate the results for comparison.

For all three subtasks, the proposed systems achieved micro  $F_1$ -scores of 0.492, 0.091, and 0.446 on the test set, respectively. The results of all models exceeded the baseline. However, there is a considerable decrease compared to the scores of 0.625, 0.215, and 0.636 achieved on the dev set.

## 4 Conclusions

In this paper, we presented our system for the SemEval-2021 Task 6, the experimental results in subtasks 1 and 3 show that our proposed ALBERT-Text-CNN model and the parallel channel model achieved a good performance in the detection of persuasion techniques in texts and images.

We participated in all three subtasks and achieved the 12th, 7th, and 11th places in the test set, respectively. In a future study, to improve the generalization ability of the model, we will focus on how to deal with the problems caused by unbalanced training data.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61702443, 61966038 and 61762091.

## References

Jimmy Lei Ba and Diederik P. Kingma. 2015. [Adam: A method for stochastic optimization](#). In *Proceedings of the 3rd International Conference on Learning Representations (ICLR-2015)*, pages 1–15.

Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. [SemEval-2020 task 11: Detection of propaganda techniques in news articles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pages 1377–1414, Barcelona (online).

Jiaxu Dao, Jin Wang, and Xuejie Zhang. 2020. [YNU-HPCC at SemEval-2020 task 11: LSTM network for detection of propaganda techniques in news articles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pages 1509–1515, Barcelona (online).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-2019)*, pages 4171–4186, Minneapolis, Minnesota.

Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. [Task 6 at SemEval-2021: Detection of persuasion techniques in texts and images](#). In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, SemEval ’21, Bangkok, Thailand.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. [Deep residual learning for image recognition](#). In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR-2016)*, pages 770–778.

Yoon Kim. 2014. [Convolutional neural networks for sentence classification](#). In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*, pages 1746–1751, Doha, Qatar.

- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. [ALBERT: A lite BERT for self-supervised learning of language representations](#). In *Proceedings of the 8th International Conference on Learning Representations (ICLR-2020)*.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. [VisualBERT: A simple and performant baseline for vision and language](#). *CoRR*, abs/1908.03557.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Giovanni Da San Martino, Stefano Cresci, Alberto Barrón-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020. [A survey on computational propaganda detection](#). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-2020)*, pages 4826–4832.
- Andrei Paraschiv, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. [UPB at SemEval-2020 task 11: Propaganda detection with domain-specific trained BERT](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pages 1853–1857, Barcelona (online).
- Bo Peng, Jin Wang, and Xuejie Zhang. 2020. [Adversarial learning of sentiment word representations for sentiment analysis](#). *Information Sciences*, 541:426–441.
- Karen Simonyan and Andrew Zisserman. 2015. [Very deep convolutional networks for large-scale image recognition](#). In *Proceedings of the 3rd International Conference on Learning Representations (ICLR-2015)*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NIPS-2017)*, pages 5998–6008.
- Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2019. [Investigating dynamic routing in tree-structured LSTM for sentiment analysis](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP-2019)*, pages 3432–3437, Hong Kong, China.
- Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2020. [Tree-structured regional cnn-lstm model for dimensional sentiment analysis](#). *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:581–591.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP-2020)*, pages 38–45.
- Li Yuan, Jin Wang, and Xuejie Zhang. 2020. [YNU-HPCC at SemEval-2020 task 8: Using a parallel-channel model for memotion analysis](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval-2020)*, pages 916–921, Barcelona (online).

# LT3 at SemEval-2021 Task 6: Using Multi-Modal Compact Bilinear Pooling to Combine Visual and Textual Understanding in Memes

Pranaydeep Singh and Els Lefever

LT3, Language and Translation Technology Team

Department of Translation, Interpreting and Communication – Ghent University

Groot-Brittanniëlaan 45, 9000 Ghent, Belgium

pranaydeep.singh@ugent.be, els.lefever@ugent.be

## Abstract

Internet memes have become ubiquitous in social media networks today. Due to their popularity, they are also a widely used mode of expression to spread disinformation online. As memes consist of a mixture of text and image, they require a multi-modal approach for automatic analysis. In this paper, we describe our contribution to the SemEval-2021 Detection of Persuasion Techniques in Texts and Images Task. We propose a Multi-Modal learning system, which incorporates “memebeddings”, viz. joint text and vision features by combining them with compact bilinear pooling, to automatically identify rhetorical and psychological disinformation techniques. The experimental results show that the proposed system constantly outperforms the competition’s baseline, and achieves the 2nd best Macro F1-score and 14th best Micro F1-score out of all participants.

## 1 Introduction

Propaganda is a mode of communication by which the interested party pursues the aim of influencing public opinion in favour of a specific agenda or ideas. This is achieved by disseminating one-sided, biased or even fake news. With the advent of social media networks, propagandist text can reach an enormous audience. Given the overload of online text produced on a daily basis, it is not feasible to monitor this manually and researchers have started to investigate automatic methods to detect propaganda in text.

In the SemEval-2020 Task 11 on Detection of Propaganda Techniques in News Articles (Da San Martino et al., 2020), participants were asked to identify 14 different propagandist techniques in news articles. The task attracted a large interest, with 44 teams participating in the task. The best systems all used pre-trained transformers and ensemble techniques. We further refer

to Da San Martino et al. (2020b) for a comprehensive review of computational propaganda detection techniques.

More recently, internet memes have emerged as a very popular mode of expression on social networks. While memes initially seemed to be used by users of specific online communities, they have gained popularity very rapidly and are today used by a very large and varied user base. Memes can be used for very different purposes: they can be used as a form of visual rhetoric (Huntington, 2013), for online bullying or trolling (Leaver, 2013), but they can also function as a kind of persuasive device, while the intended message is wrapped in humour (Shifman, 2013). As a result, they form an interesting object of study for automatically detecting propaganda techniques.

The goal of the SemEval-2021 shared task on the detection of persuasion techniques (Dimitrov et al., 2021) is to build models for identifying rhetorical and psychological techniques that are used to influence social media users in online disinformation campaigns. This paper reports on our participation in Subtask 3, which is a multi-modal task conceived as a multi-label classification problem: given a meme, the system has to identify which of the 22 techniques are used both in the textual and visual content of the meme. To solve subtask 3, we propose a multi-modal multi-task learning system, which incorporates “memebeddings”, viz. joint text and vision features combined by means of compact bilinear pooling, to automatically identify rhetorical and psychological disinformation techniques in memes.

## 2 System Architecture

### 2.1 Task Overview

The SemEval 2021 Task 6 for detection of persuasion techniques in texts and images revolved



(a)

**Actual picture of  
the media trying to  
heal the divide in America**



(b)

Figure 1: Examples of memes where the analysis of only the text (a) or both the text and image (b) are required to automatically detect the correct persuasion technique.

around identifying 22 rhetorical and psychological disinformation techniques in internet memes. These techniques cover a wide array of phenomena like *Causal Oversimplification*, *Exaggeration/Minimisation*, *Name Calling/labeling* or *Presenting irrelevant data (red herring)*. For a full list of all categories, we refer to the task description paper (Dimitrov et al., 2021).

While some techniques in certain contexts may be accurately found just by processing the textual modality, it is very difficult to consistently identify all of the techniques without complete visual and textual context. Figure 1 shows examples of the two different cases, where in the first image, it is fairly obvious that the text contains all necessary information to predict the propaganda techniques accurately, whereas in the second meme, the textual modality is not sufficient to provide all information required to correctly predict the label. To tackle the task at hand, our approach incorporates information from both domain-related text and visual pre-training, and finally combines the two modalities using Multi-modal Compact Bilinear (MCB) Pooling (Fukui et al., 2016).

## 2.2 Proposed Models

Our multi-modal system is composed of three sub-modules:

1. **The visual pre-processor:** the visual module uses a ResNet-51 architecture (He et al., 2016) which is pre-trained to identify sub-reddits (E.g. */r/motivation*, */r/pets* */r/politics*) from around 6200 Reddit memes.
2. **The text pre-processor:** the text module uses a pre-trained BERT-large-uncased model (De-

vlin et al., 2019), fine-tuned on the PTC Corpus (Martino et al., 2020a) from the SemEval 2020 Shared Task.

3. **Integration Network:** the two sets of embeddings from the first two modules are combined with MCB pooling.

### 2.2.1 Visual Embeddings

We decided to use the Resnet-51 architecture for the Visual pre-processor. This model was trained to predict one of the 18 sub-reddits the memes were scraped from. We hypothesized that the learned embeddings are able to distinguish certain elements of the meme, since the model is forced to encode the sub-reddit it comes from, and the sub-reddits represent the genre (E.g. */r/politics*, */r/sports*) or the emotion associated with the meme (E.g. */r/motivation*, */r/dankmemes*).

### 2.2.2 Textual Embeddings

For the text pre-processor we used a pre-trained bert-large-uncased model from the HuggingFace transformers package<sup>1</sup>. We fine-tuned the model with additional linear layers for the multi-label task of predicting propaganda techniques in the PTC Corpus. BERT based fine-tuned models are often used for a lot of text classification tasks and obtain state-of-the-art performances in a large number of NLP tasks like GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016).

### 2.2.3 Combined Embeddings

We train the final model with the combined embeddings from the visual and text pre-processor,

<sup>1</sup><https://huggingface.co/transformers/>



for the final task of multi-label prediction of the 22 propaganda techniques. While the visual pre-processor and textual pre-processor become excellent feature generators individually, combining the embeddings from two modalities with different dimensions (768d for text and 1024d for images) becomes very complicated.

While a dot product will be simple and efficient to compute, it will only encode a linear mapping of features, i.e first order interactions where every visual feature only interacts with just one textual feature and not multiple features. In addition, a dot product cannot be computed with two vectors having different dimensions. A cross product on the other hand, computes the relation of every feature from one modality to every feature from the second modality. While this is closer to the representation we need, the cross product of two vectors, with 768 and 1024 dimensions respectively, will be 786,432 dimensional. To use this large a vector, the classification model would need billions of parameters, making it almost impossible to train such a model in practice.

MCB Pooling combines the computation efficiency of the dot product with the higher order representation of the cross product. It was first implemented for the task of Visual Question Answering (Antol et al., 2015), which also focuses on jointly encoding textual and visual content. It centers around constructing count sketch projections of the vectors by means of the Fast Fourier Transform (FFT) to reduce them to lower-dimensional vectors without losing a lot of information. Once the vectors are projected to lower-dimensions, computing the cross-product becomes feasible again. Pham et al. (2013) demonstrated, though, that the count sketch of the outer product of two count sketch projections is the same as the convolution of the two count sketch projections, as shown in Equation 1:

$$\Psi(x \otimes q, h, s) = \Psi(x, h, s) * \Psi(q, h, s) \quad (1)$$

where  $x$  and  $q$  are the embeddings from the textual and visual modality respectively,  $\Psi(x, h, s)$  represents the count sketch projection of a vector  $x$ , and  $*$  represents the convolution operator. Figure 2 summarizes the proposed system architecture.

### 2.3 Experimental Setup

The ResNet-51 model for the visual pre-processor was trained with Stochastic Gradient Descent and

penalized with the cross-entropy loss. For the initial state we used the model pre-trained on ImageNet, and fine-tuned by replacing the classification layer.

The BERT model for the text embeddings was fine-tuned by freezing the pre-trained model and adding a linear layer as well as a classification layer. The model was trained with the AdamW optimizer, with a rate decay of 0.01, and penalized with cross-entropy as well.

The final MCB model uses embeddings from both models and combines them into a single vector of 8000 dimensions with MCB, then passes them through two linear layers of sizes 2048 and 1024 respectively, followed by a classification layer for the multi-label output for the 22 techniques. This final model is optimized with Adam and penalized with cross-entropy as well. We used the train set released by the task organizers for training, and the development set as a validation set for optimizing hyper-parameters.

## 3 Results

Table 1 summarizes the key results of our multimodal approach. While combining the visual and textual embeddings with a simple weighted averaging consistently beats the task baselines by a significant margin, using MCB Pooling results in a considerable performance increase over weighted averaging, both for Macro and Micro F1-scores. In addition, when analyzing samples that are misclassified by the weighted averaging approach, but correctly predicted by MCB Pooling, we noticed around 40 percent of the examples required combining information from both the visual and textual modalities like Figure 1(b).

While the MCB model is able to better pool the understanding of both modalities, it still fails when a complex concept or inference is involved. Figure 3a represents a common meme format frequently found in the memes we were able to obtain from Reddit. Consequently, we expect the model has sufficiently learnt the corresponding visual features to come to a correct prediction. Figure 3b, however, requires some complex visual ideas the model has to infer in combination with the text, which it fails to do frequently. We believe that jointly training visual and textual embeddings (making the learning more coherent and not disjointed between the two modalities), instead of simply attempting to combine independent vi-

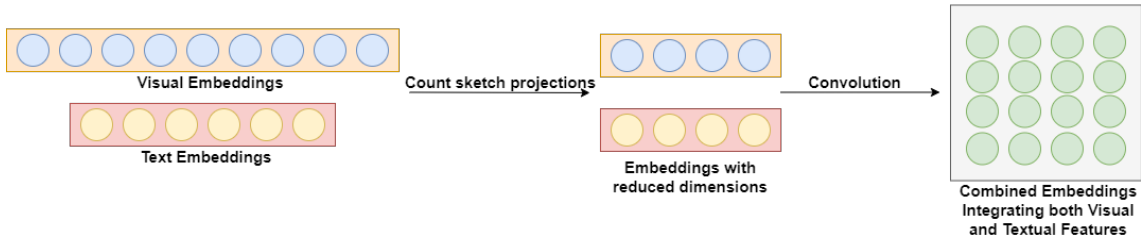


Figure 2: Conceptual overview of the system architecture

	Macro-F1	Micro-F1
Baseline	0.051	0.070
Feature Combination with Weighted Averaging	0.112	0.248
Multi-modal Compact Bilinear Pooling (our system)	<b>0.263</b>	<b>0.331</b>

Table 1: Final Micro and Macro F1-scores for SemEval 2021 Task 6 sub-task 3

Employer: \*pays minimum wage\*

Employee: \*gives minimum effort\*

Employer:



(a)



(b)

Figure 3: Examples of failed and successful classification by the MCB Pooling model. The model succeeds for standard meme templates and basic visual ideas (example *a*) but fails to understand complex visual ideas that require a joint inference from the text (example *b*)

sual and textual information, would solve this issue. However, joint training can get computationally expensive and would require a much larger dataset.

#### 4 Conclusion and Future Work

This paper presents the multi-modal approach we proposed for automatically detecting persuasion techniques in memes. As memes combine text and images to obtain the desired effect, we built a system where visual and textual embeddings are combined to classify 22 different propaganda techniques. The experimental results show that combining textual and visual embeddings by weighted averaging already beats the baseline. These results, however, are considerably improved by combining both embedding sets by means of MCB Pooling.

In future work, we will investigate how we can incorporate additional semantic information in our model. A first step could consist of integrating more explicit argumentation information into our

model. As these propaganda techniques use psychological and rhetorical techniques, we believe it might be interesting to include argumentation structures such as logical fallacies, where the reasoning is flawed, and by consequence the conclusion cannot be drawn from the premise(s) in the text. To this end, we will build on recent work on automatic fallacy detection (Habernal et al., 2017). In addition, we also aim to include automatic emotion detection features, as writers of propagandist text often use emotional language to convince their readers. Finally, we will investigate an approach to jointly train visual and textual embeddings, rather than combining separate embedding sets, as our error analysis showed that efficient analysis of memes often requires a combined approach.

## References

- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. [SemEval-2020 task 11: Detection of propaganda techniques in news articles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, Barcelona (online). International Committee for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Akira Fukui, Dong Huk Park, Daylen Yang, Anna Rohrbach, Trevor Darrell, and Marcus Rohrbach. 2016. [Multimodal compact bilinear pooling for visual question answering and visual grounding](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468, Austin, Texas. Association for Computational Linguistics.
- Ivan Habernal, Raffael Hannemann, Christian Poliak, Christopher Klamm, Patrick Pauli, and Iryna Gurevych. 2017. [Argotario: Computational argumentation meets serious games](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Copenhagen, Denmark. Association for Computational Linguistics.
- K. He, X. Zhang, S. Ren, and J. Sun. 2016. [Deep residual learning for image recognition](#). In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.
- Heidi E Huntington. 2013. Subversive memes: Internet memes as a form of visual rhetoric. *AoIR Selected Papers of Internet Research*, 3.
- Tama Leaver. 2013. Fcj-163 olympic trolls: Mainstream memes and digital discord? *The Fibreculture Journal*, (22 2013: Trolls and The Negative Space of the Internet).
- G. Da San Martino, A. Barrón-Cedeño, H. Wachsmuth, R. Petrov, and P. Nakov. 2020a. [Semeval-2020 task 11: Detection of propaganda techniques in news articles](#).
- Giovanni Da San Martino, Stefano Cresci, Alberto Barrn-Cedeño, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020b. A survey on computational propaganda detection. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 4826–4832. International Joint Conferences on Artificial Intelligence Organization. Survey track.
- Ninh Pham and Rasmus Pagh. 2013. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ questions for machine comprehension of text](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.
- Limor Shifman. 2013. Memes in a digital world: Reconciling with a conceptual troublemaker. *Journal of Computer-Mediated Communication*, 18(3):362–377.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. [GLUE: A multi-task benchmark and analysis platform for natural language understanding](#). In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

# FPAI at SemEval-2021 Task 6: BERT-MRC for Propaganda Techniques Detection

Xiaolong Hou\*, Junsong Ren\*, Gang Rao, Lianxin Jiang,  
Zhihao Ruan, Yang Mo, Jianping Shen

Ping An Life Insurance, Lt  
ShenZhen, China

{houxiaolong430, renjunsong941, raogang532, jianglianxin769,  
ruanzhihao332, moyang853, shenjianping324}@pingan.com.cn

## Abstract

The objective of subtask 2 of SemEval-2021 Task 6 is to identify techniques used together with the span(s) of text covered by each technique. This paper describes the system and model we developed for the task. We first propose a pipeline system to identify spans, then to classify the technique in the input sequence. But it severely suffers from handling the overlapping in nested span. Then we propose to formalize the task as a question answering task by machine reading comprehension (MRC) framework which achieves a better result compared to the pipeline method. Moreover, data augmentation and loss design techniques are also explored to alleviate the problem of data sparsity and imbalance. Finally, we attain the 3<sup>rd</sup> place in the final evaluation phase.

## 1 Introduction

Fake news detection has attracted the attention of many researchers. But most of the conventional fake news detection methods focus on long-form journalism, and little attention has been paid to the propaganda techniques. Memes have become the most popular type of content on social media platforms, and it can easily mislead the audience to agree with the speaker through propaganda techniques. The SemEval-2021 Task 6 focuses on detecting the use of rhetorical and psychological techniques in memes without (subtask 1, subtask 2) and with (subtask 3) visual content.

We first adopted a model to identify the span and techniques sequentially, and made many attempts to optimize the model, but the results were not satisfactory. Then we tried some end-to-end method, e.g. MRC framework.

These authors contributed equally to this work This work is licensed under a Creative Commons Attribution 4.0 International License. License details: <http://creativecommons.org/licenses/by/4.0/>

We found that the pipeline model can't handle the span overlapping issue effectively, but the MRC framework with an informative query is well performing in this scenario. In addition, the data augmentation and a carefully designed loss can alleviate the impact of data sparsity and imbalance. We attain an F1 score of 0.3974 and the 3<sup>rd</sup> place in the final evaluation phase

## 2 Related Work

There was also a related previous task on fine-grained propaganda detection (Da San Martino et al., 2019), where the participants used Transformer-style models, LSTMs and ensembles (Fadel et al., 2019; Hou and Chen, 2019; Hua, 2019). Some approaches further used non-contextualized word embeddings, e.g., based on FastText and GloVe (Gupta et al., 2019; Al-Omari et al., 2019), or handcrafted features such as LIWC, quotes and questions (Alhindi et al., 2019). Moreover, Martino et al. 2020 analysed computational propaganda detection from Text Perspective and Network Perspective, argued for the need of combined efforts blending Natural Language Processing, Network Analysis, and Machine Learning.

## 3 System Description

### 3.1 Pipeline model for Span and Technique Detection

Inspired by the method (Chernyavskiy et al., 2020) proposed for NER task, we construct a pipeline model with RoBERTa (Liu et al., 2019) as the backbone to identify spans and techniques in input sequence. Figure 1 depicts our proposed pipeline model.

#### 3.1.1 Span Identification

We treat the span identification as a binary sequence tagging task. The span identification model

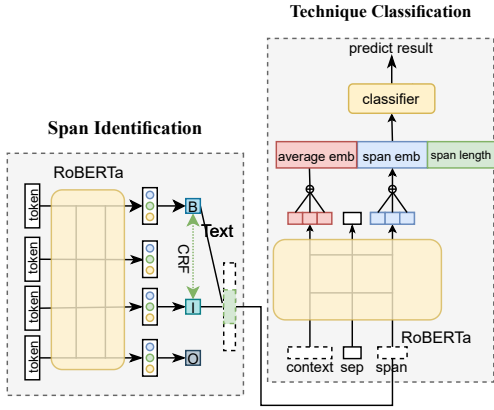


Figure 1: Pipeline Model for Span and Technique Detection

is fed with chunks of sentences encoded by Byte-Pair-Encoding (BPE) and the Conditional Random Field (CRF) layer (Lafferty et al., 2001) receives the logits for each input token, and makes a BIO prediction for the entire input sequence, finally got the spans in input sequence.

### 3.1.2 Technique Classification

We take the result of span identification model as a part of the input:  $[CLS] \langle context \rangle [SEP] \langle span \rangle$ , where  $\langle context \rangle$  is the sentence from which the span is extracted. The input of softmax layer includes three parts, (i) context embedding, is extracted from the last two layers of RoBERTa model; (ii) span embedding, is the average of span tokens embedding; (iii) span length embedding, is constructed with the length of the span, as different propaganda techniques have significant differences in span length. Finally, the model output the technique category for the given input sequence.

As the pipeline model suffers from incapable of handling overlapping issue in nested span, significantly inspired by (Li et al., 2019), we proposed to utilize the MRC framework to identify the span and corresponding techniques.

## 3.2 Span Detection as MRC

We formalize the task as a question answering task. Each span is characterized by a query, and span are extracted by answering these queries given the context. For example, the task of assigning the Name calling/Labeling to "CALM DOWN [LITTLE TRUMP HATER]\n FOUND YOUR BINKY\n" is formalized as answering the question "Find the words and phrases with strong emotional implications (either positive or negative) that in-

fluence an audience". This strategy naturally tackles the span overlapping issue in nested span: the detection of different spans that overlap requires answering different independent questions.

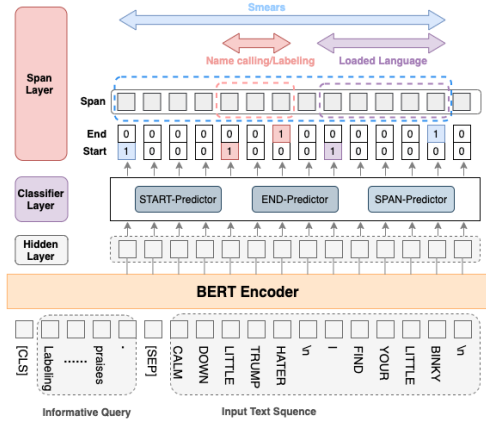


Figure 2: BERT-MRC Span Detection Model

### 3.2.1 BERT Encoder

Given the question  $q_y$ , we need to extract the text span  $x_{start,end}$  from input text sequence  $X = \{x_1, x_2, \dots, x_n\}$  given  $q_y$  using MRC frameworks. We use BERT as encoder with the concatenated informative query  $q_y$  and input sequence  $X$  as input by adding special token [CLS] and [SEP], and get the whole representation matrix  $H$  from BERT.

$$H = BERT([q_y, X]) \quad (1)$$

### 3.2.2 START-Predictor

The START-Predictor output the probability of each token being a start token of a span, and  $E_{start}$  is the parameter to learn:

$$P_{start} = softmax(H \cdot E_{start}) \quad (2)$$

### 3.2.3 END-Predictor

The END-Predictor output the probability of each token being an end token of a span,  $E_{end}$  is the parameter to learn:

$$P_{end} = softmax(H \cdot E_{end}) \quad (3)$$

### 3.2.4 SPAN-Predictor

As there could exist more than one span in given input sequence, we need to predict multiple start token and end token. Deciding which start-end pair that consist of continuous token sequence between start token and end token. We get the possible start token indexes  $I_{start}$  and end token indexes  $I_{end}$  by applying  $argmax$  on  $P_{start}$  and  $P_{end}$ . Each  $i$  in

$I_{start}$  and  $j$  in  $I_{end}$  construct a continuous token sequences  $x_{i,j}$  with constriction  $i \leq j$ . A binary classifier is used to compute the probability of  $x_{i,j}$  as a valid span, and  $E_{span}$  is the parameter to learn:

$$I_{start} = \operatorname{argmax}(P_{start}) \quad (4)$$

$$I_{end} = \operatorname{argmax}(P_{end}) \quad (5)$$

$$p_{i,j} = \operatorname{sigmoid}(E_{span} \cdot [E_i; E_j]) \quad (6)$$

### 3.2.5 Train

During the training stage, input sequence  $X$  is with two label sequence  $Y_{start}$  and  $Y_{end}$ , representing the ground-truth label of each token  $x_i$  being the start index and end index.  $Y_{start,end}$  the ground-truth span. And the loss for each predictor as follows:

$$L_{start} = \operatorname{CrossEntropy}(P_{start}, Y_{start}) \quad (7)$$

$$L_{end} = \operatorname{CrossEntropy}(P_{end}, Y_{end}) \quad (8)$$

$$L_{span} = \operatorname{CrossEntropy}(P_{span}, Y_{span}) \quad (9)$$

The model is end-to-end fine-tuned by minimizing the loss as follows:

$$L = L_{start} + L_{end} + L_{span} \quad (10)$$

### 3.2.6 Predict

For every example to be predicted, we duplicate the example 20 times and pair each example with one technique description as model input. The model output spans detected for each example, and techniques can be mapped from those examples with spans detected.

## 4 Experiment and Result

We use the provided training dataset for training, and evaluate the model on the development dataset during evaluation phase. We implemented several experiments to explore the effect of different methods. The following is the detail for each experiment.

### 4.1 Data augmentation

As there are only 290 records in the training dataset, we explored two data augmentation methods to increase the amount of relevant data. We did the data augmentation based on PTC corpus<sup>1</sup>. And Enlarge Training Dataset get a better result.

<sup>1</sup><https://propaganda.qcri.org/semEval2020-task11/>

Data augmentation	F1
Two-Stage Fine-Tune	0.19322
Enlarge Training Dataset	<b>0.21053</b>
Train Dataset	0.19073

Table 1: Result for Data Augmentation.

#### 4.1.1 Two-Stage Fine-Tune

Train a best-performing model using BERT as backbone with PTC corpus, then finetune the trained model on the training dataset. Table 1 shows the result of this method.

#### 4.1.2 Enlarge Training Dataset

Train a best-performing model using BERT as backbone with training dataset and annotate PTC corpus automatically. After that we train the second model using the filtered samples according to annotated labels and training dataset. Table 1 shows the result of this method.

### 4.2 Loss setup

This task is faced with serve data imbalance issue: the top 6 technique category examples account for 78%, which is significantly outnumber the reset of 14 technique category examples, leading to huge number of top 6 examples overwhelms training. In order to remit the impact of imbalance issue, we tried different loss functions. Table 2 shows the result of different loss function. From the result we can see that Asymmetric Loss improved the result over other loss function.

#### 4.2.1 Focal Loss

By setting  $\gamma \geq 0$  in Eq.12, the contribution of easy samples can be down-weighted in the loss function, enabling to focus more on harder samples during training(Lin et al., 2018).

$$L = -yL_+ - (1 - y)L_- \quad (11)$$

$$\begin{cases} L_+ = (1 - p)^\gamma \log p \\ L_- = p^\gamma \log (1 - p) \end{cases} \quad (12)$$

#### 4.2.2 Asymmetric Loss

Asymmetric loss can perform hard thresholding of very easy samples, meaning fully discard negative samples when their probability is low enough(Ben-

Loss Function	F1
CrossEntropy	0.19073
Focal Loss	0.20453
Asymmetric Loss	<b>0.20817</b>

Table 2: Result for Different Loss.

Model	F1
BERT-MRC <sub>neg=2</sub>	0.2986
BERT-MRC <sub>neg=4</sub>	0.3665
BERT-MRC <sub>neg=10</sub>	0.3463
BERT-MRC <sub>neg=4.DA</sub>	<b>0.3779</b>
BERT-MRC	0.2815

Table 3: Results for MRC-BERT.

Baruch et al., 2020).

$$L = -yL_+ - (1 - y)L_- \quad (13)$$

$$p_m = \max(p - m, 0) \quad (14)$$

$$ASL = \begin{cases} L_+ = (1 - p)^{\gamma_+} \log p \\ L_- = p_m^{\gamma_-} \log(1 - p_m) \end{cases} \quad (15)$$

Where the probability margin  $m \geq 0$  is a tunable hyperparameter.

### 4.3 BERT-MRC Span Detection

In the training dataset, some examples do not contain any technique. We duplicate each example 20 times and pair same example with different technique description as model input. The example paired with the technique it contained is treated as positive examples and others as negative examples. We also tried to sample different number of negative examples. Table 3 shows the result of different sampling strategy. When sample 4 negative examples, the model BERT-MRC<sub>neg=4.DA</sub> achieve the best result. The performance decrease when sample more negative examples, which may be caused by introducing too much noise into the model. The model is trained with hyperparameter learning rate  $5e-5$ , batch size 8, max sequence length 128, and bert-base-cased as the backbone.

## 5 Conclusion

We create a pipeline model and an end-to-end MRC model to identify the span and techniques. We attain an F1 score of 0.3974 and the 3<sup>rd</sup> place at in final evaluation phase with BERT-MRC<sub>neg=4.DA</sub> model. Our implementation shows that 1) reformulating the task into an MRC task to detect the

overlapping span is effective; 2) data augmentation is about to increase model generalization ; 3) careful loss design can alleviate the effect of data imbalance.

## References

- Hani Al-Omari, Malak Abdullah, Ola AlTiti, and Samira Shaikh. 2019. [JUSTDeep at NLP4IF 2019 task 1: Propaganda detection using ensemble deep learning models](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 113–118, Hong Kong, China. Association for Computational Linguistics.
- Tariq Alhindi, Jonas Pfeiffer, and Smaranda Muresan. 2019. [Fine-tuned neural models for propaganda detection at the sentence and fragment levels](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 98–102, Hong Kong, China. Association for Computational Linguistics.
- Emanuel Ben-Baruch, Tal Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. 2020. [Asymmetric loss for multi-label classification](#).
- Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. [Aschern at SemEval-2020 task 11: It takes three to tango: RoBERTa, CRF, and transfer learning](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1462–1468, Barcelona (online). International Committee for Computational Linguistics.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. [Findings of the NLP4IF-2019 shared task on fine-grained propaganda detection](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 162–170, Hong Kong, China. Association for Computational Linguistics.
- Ali Fadel, Ibraheem Tuffaha, and Mahmoud Al-Ayyoub. 2019. [Pretrained ensemble learning for fine-grained propaganda detection](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 139–142, Hong Kong, China. Association for Computational Linguistics.
- Pankaj Gupta, Khushbu Saxena, Usama Yaseen, Thomas Runkler, and Hinrich Schütze. 2019. [Neural architectures for fine-grained propaganda detection in news](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 92–97, Hong Kong, China. Association for Computational Linguistics.

- Wenjun Hou and Ying Chen. 2019. [CAUnLP at NLP4IF 2019 shared task: Context-dependent BERT for sentence-level propaganda detection](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 83–86, Hong Kong, China. Association for Computational Linguistics.
- Yiqing Hua. 2019. [Understanding BERT performance in propaganda analysis](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 135–138, Hong Kong, China. Association for Computational Linguistics.
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. [Conditional random fields: Probabilistic models for segmenting and labeling sequence data](#). pages 282–289.
- Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2019. [A unified MRC framework for named entity recognition](#). *CoRR*, abs/1910.11476.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. [Focal loss for dense object detection](#).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#).
- Giovanni Da San Martino, Stefano Cresci, Alberto Barron-Cedeno, Seunghak Yu, Roberto Di Pietro, and Preslav Nakov. 2020. [A survey on computational propaganda detection](#).



# NLPIITR at SemEval-2021 Task 6: RoBERTa Model with Data Augmentation for Persuasion Techniques Detection

Vansh Gupta

Indian Institute of Technology, Delhi  
vansh.g0108@gmail.com

Raksha Sharma

Indian Institute of Technology, Roorkee  
raksha.sharma@cs.iitr.ac.in

## Abstract

This paper describes and examines different systems to address Task 6 of SemEval-2021: *Detection of Persuasion Techniques In Texts And Images, Subtask 1*. The task aims to build a model for identifying rhetorical and psychological techniques (such as causal oversimplification, name-calling, smear) in the textual content of a meme which is often used in a disinformation campaign to influence the users. The paper provides an extensive comparison among various machine learning systems as a solution to the task. We elaborate on the pre-processing of the text data in favor of the task and present ways to overcome the class imbalance. The results show that fine-tuning a RoBERTa model gave the best results with an F1-Micro score of 0.51 on the development set.

## 1 Introduction

The purpose of this task was to identify propaganda in electoral campaigns where memes are one of the most popular types of content used to carry out disinformation campaigns. They are most effective on social media platforms since they can easily reach many users. SemEval-2021 task 6 (Dimitrov et al., 2021) refers to propaganda whenever information is purposefully shaped to foster a predetermined plan. Propaganda uses psychological and rhetorical techniques to achieve its objectives. Such techniques include the use of logical fallacies and appealing to the emotions of the audience. Logical fallacies are usually hard to spot since the argumentation, at first sight, might seem correct and objective. However, a careful analysis shows that the conclusion cannot be drawn from the premise without the misuse of logical rules. Another set of techniques uses emotional language to induce the audience to agree with the speaker only based on the emotional bond that is being created, provok-

ing the suspension of any rational analysis of the argumentation.

The Oxford dictionary defines a meme as ‘An image, video, piece of text, etc., typically humorous in nature, that is copied and spread rapidly by internet users, often with slight variations’. They generally consist of an image superimposed with text. The role of the image in a deceptive meme is either to reinforce/complement a technique in the text or to convey one or more persuasion techniques. A total of 3 subtasks were defined: For subtasks 1 and 2, only the meme’s textual content was given. The former was a multilabel classification problem while the latter was a multilabel sequence tagging problem, each with 20 labels of techniques. For subtask 3, both textual and visual content of the memes were given. The goal was to identify which of the techniques are used for a given text/image. In this paper, we describe a system to address the subtask 1 for which, a training dataset of 600 sentences with their labels was given. The test set consists of over 200 sentences. We evaluated the performance of all the models using the Micro and Macro F1 scores.

We make use of 13 different models to approach the problem at hand, experimenting with different data preprocessing techniques to finally document 23 sets of results. We found that strong results can be achieved by fine-tuning a pre-trained RoBERTa language model. However, we also discovered that some other models, when used as part of a voting classifier, gave decent results without requiring resources and time needed to train a RoBERTa model.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 talks about the methodology. Section 4 describes the dataset for the task. Section 5 provides the experimental setup. Section 6 presents the systems implemented to address the task. Section 7 shows

the results and Section 8 concludes the paper.

## 2 Related Work

The closest research to persuasion detection in Computer Science is perspective detection. Lin et al., (2006) discussed different types of machine-learning classifiers such as Naïve Bayes and Support Vector Machines for the perspective detection. Another topic, closely knitted with the detection of persuasion techniques is hate speech detection. Sood et al., (2012) worked on detecting personal insults, profanity, and user posts characterized by malicious intent. Xiang et al., (2012) focused on vulgar language and profanity-related offensive content. Xu et al., (2012) further look into jokingly formulated teasing in messages that represent bullying episodes.

Burnap and Williams, (2014) specifically looked into othering language, characterized by an ‘us’ and ‘them’ dichotomy in racist communication. Approaches to detecting hate speech on Twitter using convolutional neural networks and convolution-GRU-based deep neural networks are discussed in Gamback and Sikdar, (2017) and Zhang et al., (2018) respectively.

Persuasion technique detection is a multi-label classification problem. Many attempts have been made in the literature to make a multi-label classification system (Spyromitros et al., 2008; Elisseeff et al., 2001; Zhang and Zhou, 2006; Curram and Mingers, 1994; Huang, 1988; Crammer and Singer, 2003).

## 3 Method

In this paper, we make use of a one-versus-rest strategy to split a multi-label classification dataset into binary classification problems, splitting the data into one binary dataset for each class (Hastie et al., 1998), called ‘pairwise coupling’. We present results with various approaches for the persuasion techniques detection in the text data. We make use of pre-defined language models and tune the hyperparameter for best performance on the dataset. Soft voting classifier and hard voting classifier (Zhou et al., 2002) are deployed to use traditional machine learning models for the task. A soft voting classifier uses the average of probabilities for each class to classify the test instance, whereas, in hard voting, the classifier assigns the class that was voted by a majority of the classifiers. In our case, since we implemented our own voting classifier, we did not

always go with the majority but experimented with different threshold values. In some cases, we gave more weightage to the models which gave more promising results. Various models, such as Logistic Regression, Naive Bayes, SVM, RNN, BiLSTM, BERT, naming a few, were implemented, which are discussed in the later sections. RoBERTa model performed the best for the persuasion techniques detection task on the text data (Section 7).

## 4 Dataset

In this paper, we provide systems to address subtask-1. The input data for subtask-1 is the text extracted from the memes. The training, the development, and the test sets for subtask-1 were distributed as JSON files.

An object of the JSON has the following type:

```
{
  "id": "125"
  "labels": [
    "Loaded Language",
    "Name calling/Labeling"
  ]
  "text": "I HATE TRUMP \n\nMOST TERROR-IST DO"
}
```

where

- id is the unique identifier of the example across all three tasks
- text is the textual content of the meme, as a single UTF-8 string.
- labels is a list of techniques used in the text (Written below). In this case, two techniques were spotted: Loaded Language and Name calling/Labeling.

Below are the technique names with their frequency in the training dataset.

- Appeal to authority: 13
- Appeal to fear/prejudice: 43
- Black-and-white Fallacy/Dictatorship: 18
- Causal Oversimplification: 27
- Doubt: 34
- Exaggeration/Minimisation: 52
- Flag-waving: 27
- Glittering generalities (Virtue): 32
- Loaded Language: 313
- Misrepresentation of Someone’s Position: 3
- Name calling/Labeling: 188

Number of labels	Number of sentences
0	161
2	119
3	90
4	47
5	11
6	2
8	1

Table 1: Number of labels for number of sentences

- Obfuscation, Intentional Vagueness, Confusion: 4
- Presenting Irrelevant Data (Red Herring): 3
- Reductio ad hitlerum: 9
- Repetition: 8
- Slogans: 84
- Smears: 168
- Thought-terminating cliché: 20
- Whataboutism: 40
- Bandwagon: 2

Table 1 reports the number of sentences with multiple (or no) labels. For example, the first row suggests that there were 161 sentences in the dataset with no labels.

## 5 Experimental Setup

We train a wide range of different models for the task. As discussed in Section 3, we used both a soft voting classifier and a hard voting classifier on the traditional machine learning models for better performance. However, it should be noted that a soft voting classifier cannot be used in all the models, and accordingly, models were categorized. In order to assign different weights to different models, as per the individual performance, we used our own version of the hard voting classifier by scikit-learn (Pedregosa et al., 2011). Most of the models that we implemented support a multi-class classification where one assumes that each sample is assigned to one and only one label. However, since we were in a multi-label classification scenario, where we were required to assign a set of target labels, we wrapped these models in a OneVsRestClassifier. We converted the multi-label classification problem into a series of binary classification problems assuming that there was not any underlying correlation between any two labels and they were mutually

exclusive. Below is the list of the machine learning and deep learning approaches that we implemented to build the systems to identify persuasive techniques.

- Logistic Regression (LR), a meta-model trained using the stochastic average gradient solver.
- Standard Naive Bayes (NB) and **Support Vector Machine (SVM)** from scikit-learn library in python.
- Linear Support Vector Classification (SVC) to return a *best fit* hyperplane that divides, or categorizes the data.
- Bernoulli Naive Bayes, a variant of Naive Bayes, using learning rate of 0.12.
- Ridge classifier with 0.01 as the tolerance for the stopping criteria.
- SGDClassifier (SDGC) with added *L2* regularization to prevent overfitting and the hinge loss function.
- Passive Aggressive Classifier (PAC) with 50 as the cap for the number of iterations the model makes over the data.
- Random Forest Classifier (RFC) which creates a set of decision trees from a randomly selected subset of the training set.
- Multi-layer Perceptron (MLP) regressor with 1 hidden layer of size 8, followed by 2 of size 16 and finally another of size 4.
- Long Short-Term Memory Network (LSTM) (Hochreiter and Schmidhuber, 1997).
- Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018).
- A Robustly Optimized BERT Pretraining Approach (RoBERTa) (Liu et al., 2019).
- A Deep Neural Network model that used a vocabulary size of 10,000; a batch size of 32; and was trained over 6 epochs. The system consisted of an embedding layer of size 128, followed by a 1D Convolution layer with 32 filters and *same* padding. Further, we used a max pooling layer of pool size 2 and an LSTM layer with 64 units and Dropout regularization of 0.2. Finally, 2 dense layers with 64 units and 1 unit respectively and activations *ReLU* and *sigmoid* were used.

**Evaluation Measures:** The given evaluation measure for the task was Micro-F1. We also report Macro-F1 in this paper as it is more favorable for multi-class classification problems.

F1-score is the harmonic mean of Precision and Recall

**Precision:** It is the fraction of relevant instances among all retrieved instances

**Recall:** It is the fraction of retrieved instances among all relevant instances

First, we independently find the F1 Score for each label, and then, to aggregate these F1-scores into a single F1-score, two ways can be used.

- **Macro F1-Score:** Simply average all the F1-Scores and calculate a mean F1-Score.
- **Micro F1-Score:** Instead of calculating each label's F1-Score, derive the F1-Score by calculating Precision and Recall by summing all the true positives, false positives, and false negatives of the system for different labels.

## 6 System Description

We split the training data into a training and validation set in the ratio 9 : 1, *i.e.*, 90% of the corpus was used for training while the remaining 10% was part of the validation set. The preprocessing (Section 6.3) for all the models was the same and once the best model was found, it was retrained including the validation set for training as well. The trained model was then used to predict labels for the Development set. The model yielding the best accuracy on the earlier chosen validation set was used for the submission of the Development set. We followed a similar approach for the submission of the results in the task on the test set using RoBERTa model. We secured 15<sup>th</sup> rank in the task.

### 6.1 Initial Setup

The labels were in the form of a string under the column 'labels'. For easier manipulation, these were first one hot encoded into multiple labels using ',' as a separator. It should be noted that multiple columns for the same label (Owing to white space before the comma in a few cases) had to be merged.

### 6.2 Data Augmentation

We observed a lack of data along with an internal data bias. To overcome this issue, we incorporated different data augmentation techniques. While a few of the techniques we employed for data augmentation boosted the performance, many of them

just confused the model and decreased the validation set accuracy. The techniques which were discarded for making no significant improvement on the results are as follows:

- **Random Swap:** Choose two words in the sentence at random, and swap their positions.
- **Synonym Replacement:** Choose a few words from the sentence at random, and replace them with their synonyms.
- **Random Deletion:** Randomly delete a word if a uniformly generated number between 0 and 1 is smaller than a pre-defined threshold.
- **Random Insertion:** Find a synonym of a word chosen at random from the sentence and insert it into a random position in the sentence.

However, *Back Translation* which is a classic data augmentation technique gave positive results. This method translates the text data to some language and then translates it back to the original language. This can help to generate textual data with different words while preserving the context of the text data. In our model, we made use of the Language translation API provided by Google Translate. We ran each text sentence through 5 different translations. Each series consisted of translation from English to another language of the similar scripts followed by another language that required transliteration and then back to English. We used this with 10 different languages (other than English) to create maximum diversity in the corpus. After we had generated a much larger corpus than before, there was still a serious problem of data bias in most labels since there were more zeroes (in one hot encoding) than ones. For that, apart from the original corpus, we only took those text samples out of the augmented data, in which that label was present. Compared to the previous training dataset, upon using this dataset, the performance (Micro F1 score) of the model on the development set rose for a few models while it fell for others. For the latter, we believe that the decrease in data-size was under-compensated by the decrease in the problem of bias.

### 6.3 Preprocessing

Since the corpus sentences were texts extracted from memes, we expected them to contain some patois and other contractions. Consequently, we started our preprocessing by substituting conjunctions with the complete words, and some of the more commonly used lingos with their literal and

System Name	Drop	Threshold	Aug F1-Micro	F1-Micro	Aug F1-Macro	F1-Macro
<b>RoBERTa</b> (Dev Set)	×		<b>0.50909</b>		<b>0.34419</b>	-
All models together	×	0	0.44118	0.38450	0.20520	0.18398
	×	1	0.40495	0.34520	0.18473	0.15812
	×	2	0.37919	0.30891	0.17666	0.12938
	×	3	0.35689	0.26981	0.16338	0.11112
	×	4	0.33835	0.26818	0.13557	0.10442
All models together	✓	0	0.34404	—	0.20039	—
	✓	1	0.39575	—	0.21937	—
	✓	2	0.41728	—	0.22840	—
	✓	3	0.43426	—	0.23603	—
	✓	4	0.44021	—	0.23809	—
Ridge Classifier	×	—	0.38983	0.29870	0.14469	0.12775
Rand Forest Clf	×	—	0.37209	0.17021	0.15921	0.07692
Naive Bayes Clf	×	—	0.35805	0.27189	0.11108	0.08185
Logit Repr Clf	×	—	0.35448	0.27865	0.12454	0.07897
SGDC Classifier	×	—	0.34752	0.32150	0.17489	0.14933
BNB Classifier	×	—	0.34520	0.26957	0.15900	0.11105
SVM Classifier	×	—	0.34409	0.29857	0.17228	0.13124
PAC Classifier	×	—	0.34295	0.30435	0.17311	0.15109
MLP Regressor	×	—	0.32975	0.28200	0.14570	0.14747
Deep Neural Network	×	0.2	0.31390	0.32819	0.14713	0.12470
	×	0.3	0.32049	0.32283	0.14894	0.12299
	×	0.4	0.32137	0.31855	0.14894	0.11849
	×	0.5	0.31360	0.31919	0.13993	0.11866

Table 2: Results with various systems

legal English counterparts.

After that, we proceeded by using custom made functions for subjecting the sentences to lowercasing and removing most punctuation marks or any kind of special character which do not include any valuable information for text classification.

Next, we take care of the stopwords, which are a set of frequently used words (often short) and their removal is critical because, in their absence, we can focus on the essential words to the context of the sentence more. Therefore, we remove all the stop-words present in the text sentences using the default set of stop-words that can be downloaded from the NLTK library (Bird et al., 2009).

Next, we moved to Lemmatization. Stemming is another widely used technique for a similar purpose where one chops off its inflections and keeps what hopefully represents the main essence of the word. However, we observed that lemmatization gave better results on the provided data. Instead of chopping words off (truncating the word), lemmatization relies on a linguistic knowledge base like

*WordNet* (An extensive database of English which superficially resembles a thesaurus, in that it groups words based on their meanings) to obtain the correct base forms of words.

## 7 Results and Analysis

Table 2 presents the F1-Micro and F1-Macro obtained with various systems on the test data for the subtask-1 of Task-6 after training on both the augmented data and just the original base data. The scores for the models trained on the augmented data use the prefix *Aug* in the column name. The models with the *Drop* column marked *X*, were trained on the complete augmented data. In contrast, while training the ones containing a *✓*, we took only those augmented sentences that contained the label we were training the model for. This was done to cope with the data bias problem as discussed in Section 6. In the machine learning models, the *Threshold* column contains an integer value which is the threshold for the hard voting classifier *i.e.*, the minimum number of models predicting a technique

to be associated with a sentence. However, in the case of the Deep Neural Network, the *Threshold* column contains the probability of the prediction of a particular technique.

The system ‘All models together’ deployed an ensemble of traditional machine learning models, as listed in the Section 5. From table 2, we unsurprisingly infer that *RoBERTa* gave the best results for the persuasion techniques classification on the text data. Even so, there are quite a few takeaway points one can conclude from the same.

Firstly, even though there is a visible gap between RoBERTa and the rest of the models, the ensemble of models was almost 200 times faster on a CPU compared to RoBERTa, which was trained on a GPU. This indicates that the former is a superior choice when quick NLP predictions are required using low-end systems.

Next, we observe the effect of tackling the data bias technique by choosing some specific augmented sentences for the training corpus. In ‘All models together’, even though a better result was reported by the system without the drop and the threshold value 0, overall, dropping the sentences gave better performance for different threshold values. One can note that the latter system even gave better F1-Macro score universally for all thresholds.

Finally, we adjudge a rather interesting observation. Most of the systems comprehensively work far better with the augmented data, but the Deep Neural Network, which contained 1.5 Million trainable parameters, showed a startling outcome. The score obtained after training on the base corpus was better than the one obtained after training on much larger augmented data. Repeated back-translation might be the reason that we fed the similar type of data instances to the system. Thus, the model was not able to learn enough, giving poor accuracy and faulty predictions. We didn’t see the similar observation in traditional machine learning models or even MLP regressors because these models use fewer parameters.

## 8 Conclusion

Persuasion techniques detection is a multi-label classification problem. This paper presents and analyzes 13 machine learning and deep learning-based systems for persuasion techniques detection in the text data. Persuasion data has an extensive range of techniques as labels (20 in our case) with

high-class imbalance. We observe that the data augmentation technique, *i.e.*, Back Translation, helps overcome class imbalance and produces more robust systems. Besides, we present the essential data preprocessing for the task. Results show that RoBERTa, a deep neural language model, outperformed other systems by a significant margin. Fine tuning the RoBERTa model on the training data captures the sensitive features for persuasion techniques identification.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Peter Burnap and Matthew Leighton Williams. 2014. Hate speech, machine classification and statistical modelling of information flows on twitter: Interpretation and communication for policy decision making.
- Koby Crammer and Yoram Singer. 2003. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3(Feb):1025–1058.
- Stephen P Curram and John Mingers. 1994. Neural networks, decision tree induction and discriminant analysis: An empirical comparison. *Journal of the Operational Research Society*, 45(4):440–450.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval ’21*, Bangkok, Thailand.
- André Elisseeff, Jason Weston, et al. 2001. A kernel method for multi-labelled classification. In *NIPS*, volume 14, pages 681–687.
- Björn Gambäck and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hate-speech. In *Proceedings of the first workshop on abusive language online*, pages 85–90.
- Trevor Hastie, Robert Tibshirani, et al. 1998. Classification by pairwise coupling. *Annals of statistics*, 26(2):451–471.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

- William Y Huang. 1988. Neural net and traditional classifiers. In *Neural Information Processing Systems: Proceedings of a conference held in Denver, Colorado, November 1987*, volume 1, page 387. Springer Science & Business Media.
- Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. 2006. [Which side are you on?](#) In *Proceedings of the Tenth Conference on Computational Natural Language Learning - CoNLL-X '06*. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. 2012. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63(2):270–285.
- Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas. 2008. An empirical study of lazy multilabel classification algorithms. In *Hellenic conference on artificial intelligence*, pages 401–406. Springer.
- Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984.
- Jun-Ming Xu, Kwang-Sung Jun, Xiaojin Zhu, and Amy Bellmore. 2012. Learning from bullying traces in social media. In *Proceedings of the 2012 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 656–666.
- Min-Ling Zhang and Zhi-Hua Zhou. 2006. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10):1338–1351.
- Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *European semantic web conference*, pages 745–760. Springer.
- Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. 2002. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263.

# LeCun at SemEval-2021 Task 6: Detecting Persuasion Techniques in Text Using Ensembled Pretrained Transformers and Data Augmentation

Dia Abujaber\*    Ahmed Qarqaz\*    Malak Abdullah

Jordan University of Science and Technology

Irbid, Jordan

daabujaber17, afalqarqaz17@cit.just.edu.jo

mabdullah@just.edu.jo

## Abstract

This paper presents one of the top systems for the SemEval-2021 task 6 (Dimitrov et al., 2021), “detection of persuasion techniques in text and images”. The proposed system, LeCun, targets subtask-1 for detecting propaganda techniques based on the textual content of a meme. We have used an external dataset from a previous relevant SemEval competition (Martino et al., 2020). We also have articulated another dataset using data-augmentation techniques. The final proposed model consisted of 5 ensemble transformers (four RoBERTa models and one DeBERTa), each trained on either a different dataset or pre-processing. Apparently, ensembling models trained on different datasets improve performance more than ensembling models trained on the same dataset/preprocessing. Also, it is obvious, fine-tuning the model on the Competition dataset after training it for a few epochs on the other datasets would improve the f1-micro up to 0.1 additional scores. The final model achieved an f1-micro score of 0.512 on the test dataset and an f1-micro of 0.647 on the development dataset.

## 1 Introduction

The definition of Memes was constantly changing since it was first conceived. But, Memes eventually got an academic definition, called an “Internet Meme”. As Davison (2012) Internet Meme can roughly be defined as “a piece of culture, typically a joke, which gains influence through online transmission”. But what makes Internet memes unique is the speed of their transmission and the fidelity of their form. Therefore the Internet meme can act as a powerful medium for persuasion techniques that preach an ideology or way of thinking. (Moody-Ramirez and Church, 2019)

On the other hand, the term “propaganda” is defined as a form of communication that employs

persuasive strategies and attempts to achieve a response that furthers the desired intent of the propagandist (Jowett and O’donnell, 2018). With the rise of social media, a new form of propaganda rises called “Computational Propaganda.” The author in (Woolley and Howard, 2017) defined Computational Propaganda as “The use of algorithms, automation, and human curation to purposefully distribute misleading information over social media networks”.

Task 6 at SemEval-2021 (Dimitrov et al., 2021), detection of persuasion techniques in text and images, defined three subtasks. The first two subtasks deal with the textual contents of memes that ask the participants to identify which of the 20 propaganda techniques are in the text. While the third subtask encourages the participants to determine which of the 22 techniques are in the meme’s textual and visual content. This paper proposes a solution for subtask1 that uses pre-trained language models to detect propaganda and possibly even identify the persuasion strategy that the propaganda sample employs.

The rest of the paper is broken down as follows. Section 2 discusses related-work to the task of propaganda identification. Section 3 provides a description of the data and the pre-processing techniques used. Section 4 describes the proposed system and architecture. Section 5 presents system analysis. Finally, the conclusion and future work are provided in Section 6.

## 2 Related Work

There have been efforts in persuasion techniques identification and classification using machine and deep learning-based approaches. The authors in (Al-Omari et al., 2019) used word embeddings with BERT (Devlin et al., 2019) and BiLSTM (Schuster

---

\* Equal Contribution



and Paliwal, 1997) for binary detection of propaganda spans. Authors in (Altitı et al., 2020) experimented with a CNN (LeCun et al., 1999), BiLSTM and BERT and showed BERT to have the best accuracy on classifying persuasion techniques in propaganda spans. Also, the authors in (Jurkiewicz et al., 2020) used a RoBERTa model (Liu et al., 2019), a class-dependent re-weighting method and used a semi-supervised learning technique of self-training and demonstrated the effects of these techniques in an ablation study. A group of researchers (Morio et al., 2020) experimented with a variety of PLMs (pre-trained language models), including BERT, GPT-2 (Radford et al., 2019), RoBERTa, XLM-RoBERTa (Conneau et al., 2019), XLNet (Yang et al., 2019) and XLM (CONNEAU and Lample, 2019). And have demonstrated that RoBERTa and XLNet generally perform better for propaganda detection.

### 3 Data Description

In this section, we describe the data and the task and the preprocessing step

#### 3.1 Data

The dataset used during our experiments has been provided by the SemEval-2021 Task 6 (Dimitrov et al., 2021). The dataset, “Competition dataset”, consists of short text samples that were extracted from Memes. We have also resorted to using an external dataset (Da San Martino et al., 2019) that is comprised of news articles with propaganda spans, “External Dataset”. To use the External dataset effectively, we needed to chop down the news articles closer to the text’s length in the current dataset and take only the text fragment that contained the propaganda and the corresponding label representing the propaganda technique in that text fragment.

#### 3.2 Data Preprocessing

Our data preprocessing pipeline consists of two components, 1) Data cleaning 2) Data augmentation. In this section, we will describe the techniques we used in each component.

##### 3.2.1 Data Cleaning

To increase performance accuracy, some data preprocessing techniques have been tested. We have

---

See [https://github.com/jasonwei20/eda\\_nlp](https://github.com/jasonwei20/eda_nlp) for the data augmentation code

experimented with typical pre-processing techniques, such as “Stop-Words Removal”, which refers to removing commonly used words (such as “the”, “a”, “an”, “in”) to eliminate noise that may otherwise hinder the model’s ability to learn and predict sequences. We have also experimented with “Stemming” which refers to the process of reducing inflection in words (e.g., connect, connected, connection) to their root form (e.g., connect). The specific Stemming algorithm that was used is Porters Algorithm (Porter, 1980).

##### 3.2.2 Data Augmentation

We experimented with Data Augmentation (Wei and Zou, 2019). This is the process of using the original given data to produce more data to increase the given dataset size. Data Augmentation has been proved to be useful when dealing with small datasets. Although this technique is more prevalent in computer vision tasks, there are some versions of the technique that are specifically tailored to work with text data as described at (Wei and Zou, 2019). These techniques include Synonym Replacement, Random Insertion, Random Swap, Random Deletion, Back-translation. Table 1 shows examples on generating data using data-augmentation. This was done by using the “Easy Data Augmentation” library (Wei and Zou, 2019).

Back-translation was only applied on the Competition dataset and the other four techniques on the External dataset. For each sentence in the External dataset, 0.1 percent of Synonym Replacement, Random Insertion, Random Swap, and Random deletion was applied. We did that nine times per sentence for each sample. In the Back-translation, AWS API was used to translate the text from English to Dutch back to English. Also, from English to Dutch to Russian, back to English. For each sample, we generate two additional samples. The Competition dataset has a size of 487. After merging the External dataset and the Competition dataset, we ended up with a dataset of size 18,571. This data will be referred to as the “Competition + External Dataset”. After applying data augmentation on the Competition + External dataset, we ended up with a dataset of size 52,966. This data will be referred to as the “Augmented Dataset.”

### 4 System Description

Different model architectures have experimented with different pre-processing techniques. The final system ended up ensembling five models, each

<b>Original</b>	<i>This paper will describe our system in detecting propaganda in memes</i>
<b>Synonym Replacement</b>	<i>This theme will describe our arrangement in detecting propaganda in memes</i>
<b>Random Insertion</b>	<i>This key out paper will describe our system meme in detecting propaganda in memes</i>
<b>Random Swap</b>	<i>This paper in describe our system in detecting propaganda will memes</i>
<b>Random Deletion</b>	<i>This paper will describe system in detecting propaganda in memes</i>
<b>Back-translation</b>	<i>This document describes our Memorandum Propaganda Detection System</i>

Table 1: Generated Samples using Data-augmentation

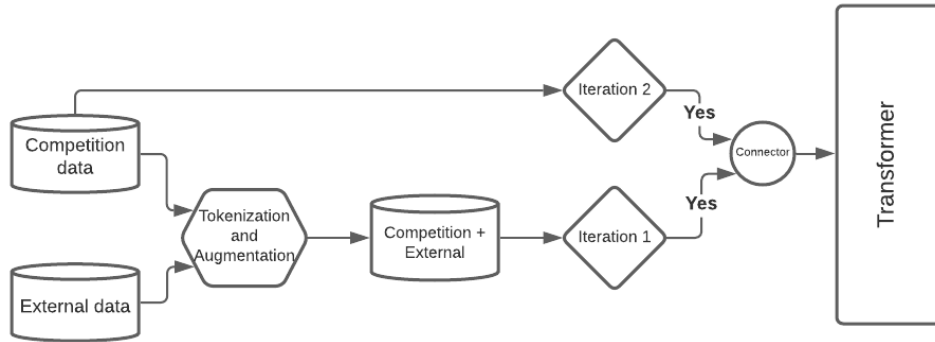


Figure 1: Approach 2 Training Schema

trained on a different approach. The ensemble model consists of 1 DeBERTa (He et al., 2021) model and 4 RoBERTa models each trained in a different approach or data pre-processing technique. We have two training approaches we used in training our models. The first one is a typical fine-tuning. The second approach consists of two iterations. In the first iteration, the model is trained on the pre-processed dataset. In the second iteration, the model from the first iteration is fine-tuned on the Competition dataset exclusively. Figure 1 demonstrates the second approach.

#### 4.1 Proposed System

The system is an ensemble model of 5 classifiers. One of them is using the DeBERTa large classifier, and the rest are RoBERTa large classifiers. Each classifier is trained on a different approach/pre-processing. For the DeBERTa large classifier, the Augmented dataset was used with the stop words removed and lowered text case. Then we trained it on the first approach for six epochs. It achieved F1 micro of 0.554 on the development set. As mentioned earlier, there are 4 RoBERTa large classifiers; the first classifier is trained on the Competition dataset and the External dataset without augmentation. We dropped samples that do not have any propaganda technique and trained the model on the first approach for four epochs. It achieved an F1 micro score of 0.550. The second RoBERTa classifier

has the same pre-processing as the first RoBERTa classifier but is trained on the second approach. It achieved an F1 micro score of 0.602 on the development set. The third RoBERTa classifier is trained on the Augmented dataset with the stop words and trained using the first approach with four epochs. It achieved F1 micro of 0.54. The fourth RoBERTa classifier is the same as the third model but fine-tuned on the Competition dataset and achieved an f1 score of 0.62. Table 2 summarizes the performance of the classifiers of LeCun’s ensemble model.

## 5 System Analysis

### 5.1 Ensemble Analysis

This section will be analyzing different combinations of the models that lead to the final proposed system. In the second approach, we noticed that fine-tuning the classifiers from the first iteration on the Competition dataset will always boost the performance up to 0.1 additional f1 micro scores on the development set. However, when it came to the ensemble model, it turned out that the ensemble model with classifiers from the second training approach doesn’t increase the overall performance, and sometimes it decreased it. Table 3 demonstrates the performance of different classifiers combinations. Ensemble (A) consists of classifier (3) with f1 micro of 0.602 and classifier (5) with f1 micro of 0.62. After the ensemble, the overall score

#	Model Type	Epochs	LR	SQM	BS	Dataset	Approach	F1-Macro	F1-Micro
1	DeBERTa Large	3	2e-06	164	8	Augmented	1	0.430	0.554
2	RoBERTa Large	2	2e-05	64	8	Competition + External	1	0.340	0.550
3	RoBERTa Large	3	2e-05	64	8	Competition + External	2	0.388	0.602
4	RoBERTa Large	4	2e-05	128	16	Augmented	1	0.358	0.540
5	RoBERTa Large	2	2e-05	128	16	Augmented	2	0.430	0.622

Table 2: Models Hyper-parameters. LR (Learning Rate), SQM (Sequence Max Length), BS (Batch Size)

	Ensemble Combination	F1 Micro	F1 Macro
A	(3)(5)	0.58	0.40
B	(4)(5)	0.59	0.40
C	(3)(4)(5)	0.62	0.41
D	(2)(3)(4)(5)	0.60	0.37
E	(1)(2)(3)	0.56	0.39
F	(1)(2)(3)(4)	0.59	0.40
G	(1)(2)(3)(4)(5)	0.64	0.44

Table 3: Performance on Different Ensemble Combinations

dropped to 0.58. However, Ensemble (A) suggests that ensembling classifier (3) and classifier (5) isn't optimal. In the final ensemble model (G), we noticed that the overall score would decrease if we removed one of these models. For example, in ensemble (F), classifier (5) was dropped, and both f1 micro and macro decreased.

## 5.2 Error Analysis

This section examines the ensemble model weaknesses to give insight on what to do next to improve the model performance. We have generated the confusion matrix and scores for each class for the test set (see Appendix). In the confusion matrices, the "None" class indicates that either the model predicted an incorrect class (not in the ground-truth labels set) or it didn't predict the correct class (in the ground-truth labels set but not in the predicted labels set). It is worth noting that the correctly classified "None" (not in the predicted labels set and not in the ground-truth labels set) is not provided in the model evaluation confusion matrix. We noticed that the model performs poorly at detecting the classes (last column) in the input test (see Figure A2). One possible explanation for this is that the model is trained on data that has a lot of samples without propaganda (count of labels = 0) (see Figure A1). In addition to that, the label matrix is sparse (zero is the dominant label in a one-hot vector). One possible solution is to remove samples with zero labels and rely on the

sparsity of vectors in detecting samples without propaganda. Another possible solution is to train a Two-Stage model, where the first stage filters out non-propaganda samples and the second stage classify the propaganda samples.

## 6 Conclusion and Future Work

In this paper, we presented our proposed system, LeCun, for detecting propaganda in contextual content. We have used an external dataset from a previous SemEval competition and performed a data-augmentation on the external dataset to expand the dataset size. We have also investigated different ensemble combinations for state-of-the-art pre-trained language models. However, there are many questions we got throughout our participation in this competition which made us curious to investigate. These questions are:- What is the influence of the data augmentation on the model performance? What is the influence of using an external dataset? How can the model weaknesses be improved? How can span identification help in improving the score of technique classification?

For future work, we will be working on answering these questions. We plan to do more in-depth experimenting with different augmentation techniques and different model architectures. We will also investigate the influence of the external dataset by training models on the competition dataset, external dataset separately and compare the final results of each.

## References

- Hani Al-Omari, Malak Abdullah, Ola AlTiti, and Samira Shaikh. 2019. [JUSTDeep at NLP4IF 2019 task 1: Propaganda detection using ensemble deep learning models](#). In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 113–118, Hong Kong, China. Association for Computational Linguistics.
- Ola AlTiti, Malak Abdullah, and Rasha Obiedat. 2020. [JUST at SemEval-2020 task 11: Detecting propa-](#)

- ganda techniques using BERT pre-trained model. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1749–1755, Barcelona (online). International Committee for Computational Linguistics.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Alexis CONNEAU and Guillaume Lample. 2019. [Cross-lingual language model pretraining](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, EMNLP-IJCNLP 2019, Hong Kong, China.
- Patrick Davison. 2012. The language of internet memes. *The social media reader*, pages 120–134.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at SemEval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. [DeBERTa: Decoding-enhanced bert with disentangled attention](#).
- Garth S Jowett and Victoria O’donnell. 2018. *Propaganda & persuasion*. Sage publications.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. [ApplicaAI at SemEval-2020 task 11: On RoBERTa-CRF, span CLS and whether self-training helps them](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1415–1424, Barcelona (online). International Committee for Computational Linguistics.
- Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. 1999. *Object Recognition with Gradient-Based Learning*, pages 319–345. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized bert pretraining approach](#).
- G. Da San Martino, A. Barrón-Cedeño, H. Wachsmuth, R. Petrov, and P. Nakov. 2020. [SemEval-2020 task 11: Detection of propaganda techniques in news articles](#).
- Mia Moody-Ramirez and Andrew B Church. 2019. Analysis of Facebook meme groups used during the 2016 US presidential election. *Social Media+ Society*, 5(1):2056305118808799.
- Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. [Hitachi at SemEval-2020 task 11: An empirical study of pre-trained transformer family for propaganda detection](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1739–1748, Barcelona (online). International Committee for Computational Linguistics.
- Martin F Porter. 1980. An algorithm for suffix stripping. *Program*.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.
- Jason Wei and Kai Zou. 2019. [EDA: Easy data augmentation techniques for boosting performance on text classification tasks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.
- Samuel C Woolley and Philip Howard. 2017. Computational propaganda worldwide: Executive summary.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. [XLNet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

## A Appendix

Label	F1-score	Precision	Recall	Support
Appeal to authority	0.0	0.0	0.0	7
Appeal to fear/prejudice	0.57	0.40	0.47	10
Black-and-white Fallacy/Dictatorship	1.0	0.29	0.44	77
Causal Oversimplification	0.0	0.0	0.0	3
Exaggeration/Minimisation	0.60	0.47	0.53	19
Flag-waving	0.40	0.33	0.36	6
Name calling/Labeling	0.64	0.55	0.59	53
Loaded Language	0.80	0.74	0.77	100
Presenting Irrelevant Data (Red Herring)	0.0	0.0	0.0	4
Reductio ad hitlerum	1.0	0.33	0.50	3
Misrepresentation of Someone's Position (Straw Man)	0.0	0.0	0.0	1
Thought-terminating cliché	0.0	0.0	0.0	6
Bandwagon	0.0	0.0	0.0	1
Doubt	0.67	0.21	0.32	28
Repetition	0.0	0.0	0.0	1
Slogans	0.2	0.05	0.08	19
Whataboutism	1.0	0.1	0.18	10
Smears	0.67	0.18	0.28	45
Glittering generalities (Virtue)	0.0	0.0	0.0	11
Obfuscation, Intentional vagueness, Confusion	0.0	0.0	0.0	1

Table 1: Classification report of the submitted system on the test set

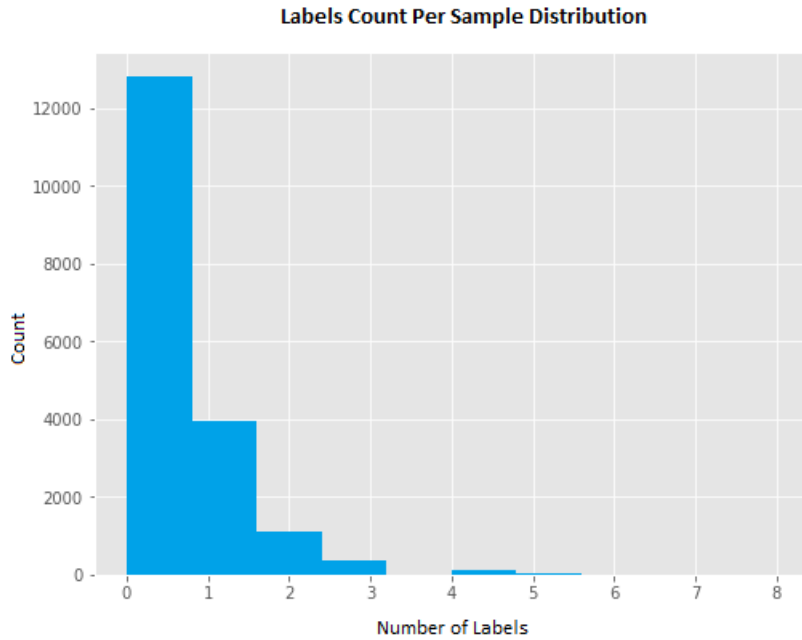


Figure A1: Labels count per sample distribution of Competition + official dataset



Figure A2: Confusion matrix of the submitted system on the test set -  $i$ -th row and  $j$ -th column entry indicates the number of samples with true label being  $i$ -th class and predicted label being  $j$ -th class

# Volta at SemEval-2021 Task 6: Towards Detecting Persuasive Texts and Images using Textual and Multimodal Ensemble

Kshitij Gupta\*    Devansh Gautam\*    Radhika Mamidi

International Institute of Information Technology Hyderabad

{kshitij.gupta, devansh.gautam}@research.iiit.ac.in,  
radhika.mamidi@iiit.ac.in

## Abstract

Memes are one of the most popular types of content used to spread information online. They can influence a large number of people through rhetorical and psychological techniques. The task, *Detection of Persuasion Techniques in Texts and Images*, is to detect these persuasive techniques in memes. It consists of three subtasks: (A) Multi-label classification using textual content, (B) Multi-label classification and span identification using textual content, and (C) Multi-label classification using visual and textual content. In this paper, we propose a transfer learning approach to fine-tune BERT-based models in different modalities. We also explore the effectiveness of ensembles of models trained in different modalities. We achieve an F1-score of 57.0, 48.2, and 52.1 in the corresponding subtasks.

## 1 Introduction

Memes are text superimposed on graphics that convey messages in the form of jokes, sarcasm, etc. In the current era of the internet and social media, they are very quick to spread. If used as a part of a disinformation campaign, it can be quite tricky to notice the agenda behind them and has the potential to influence a large mass of people without them realizing it (Muller, 2018; Tardáguila et al., 2018; Glowacki et al., 2018).

To this end, SemEval 2021 Task 6 (Dimitrov et al., 2021) focuses on identifying such persuasive techniques (Miller, 1939) in a multimodal (visual-linguistic) setting. It consists of three subtasks that enable the participants to study the problem in each modality. Only the English textual cues are used in tasks A and B, while the visual cues are also used in task C. Task B is a modification of task A which further requires predicting the spans for each identified technique as well.

\*The authors have contributed equally.



Figure 1: Sample memes demonstrating the multimodal setting

Meme classification is a multimodal problem that often requires visual and textual cues to convey a message. Memes can convey very different meanings if either of the cues is removed. A few samples are shown in figure 1 to demonstrate the importance of visual and textual cues for classification.

We experiment with BERT (Devlin et al., 2019) based unimodal models for tasks A and B. Since they are state-of-the-art models for natural language understanding, they are a good choice for understanding the complex propaganda techniques in texts. Transformers (Vaswani et al., 2017) have limited sequence length, which limits their performance on longer data, but in the case of memes, the textual content is also very limited.

For task C, we experiment with Visual-Linguistic (VL) models like UNITER (Chen et al., 2020), VisualBERT (Li et al., 2019), LXMERT (Tan and Bansal, 2019) for the cross-modal understanding of memes. We further explore the effectiveness of ensembling models trained in

different modalities.

The code for all subtasks is available at <http://github.com/kshiti98/multimodal-propaganda.git>

## 2 Background

Propaganda aims to push biased agendas to influence people’s mindsets. It is successful in achieving its goal by hiding its way through any of the numerous media platforms available in the current world. A major factor behind the success of such campaigns is the boom of the internet and social media in recent years. Another factor being the difficulty to spot such techniques manually because of the high volume of text produced and the unnoticeable nature of such content. With the recent interest of the research community in “fake news,” the detection of persuasive techniques or highly biased texts has emerged as an active research area. Some of the previous work in this direction analyzes the general pattern of propaganda (Garimella et al., 2018; Chatfield et al., 2015), performs analysis at a document level (Rashkin et al., 2017; Barrón-Cedeño et al., 2019) and a fine-grained analysis of the text (Da San Martino et al., 2019, 2020). However, most previous work analyses the techniques in a textual unimodal setting only. This work studies propaganda techniques in a new age domain like memes.

Meme classification task can be considered a combined VL multimodal problem. It is similar to the currently popular VL problems like Visual Question Answering (Antol et al., 2015), Visual Commonsense Reasoning (Zellers et al., 2019) and Visual Entailment (Xie et al., 2019), as we have to classify semantically correlated text with that of the visual content in the image. Hence a cross-modal approach under vision and text should perform better than unimodal architectures. Basic VL approaches are based on simple fusion techniques in the form of early or late fusion to correlate unimodally trained visual and textual models. However, in an ideal scenario, a multimodally trained model should be more effective in detecting persuasive techniques in memes. With the rising interest in VL problems, recent work attempts to study similar problems in a VL multimodal setting (Gomez et al., 2020; Kiela et al., 2020; Suryawanshi et al., 2020).

**Data Description** The dataset consists of 951 memes in total, which is further divided into train/

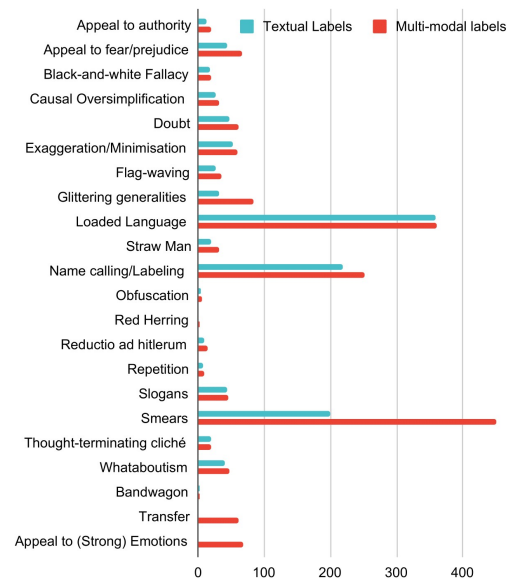


Figure 2: Data Distribution of the labels in the training set

dev/ test splits. All the tasks have the same set of memes in their training sets, but the labeling differs for each of them. For task A, only the textual cues were used to identify the techniques. For task B, the spans of each technique were further detected. For task C, more techniques were identified using visual cues.

The distribution of the labels is illustrated in figure 2. Detailed information of the dataset can be found in the task description paper (Dimitrov et al., 2021).

## 3 System Overview

Systems proposed for all the tasks use BERT-based models with task-specific modifications.

The proposed systems are explained in detail below:

### 3.1 Task A - Text Classification

We use transfer learning to fine-tune BERT-based models for this task. To fine-tune the models, we experiment with BASE and LARGE variants of BERT and RoBERTa. We attach a feed-forward network on the [CLS] token embedding with two linear layers having the model’s default dropout and *Tanh* activation layer in between. The model architecture is illustrated in figure 3a. We apply a standard Binary Cross Entropy loss to train this model with the hyperparameters mentioned in table 1. Since there is a substantial class imbalance in the dataset, we add weights to the positive samples in



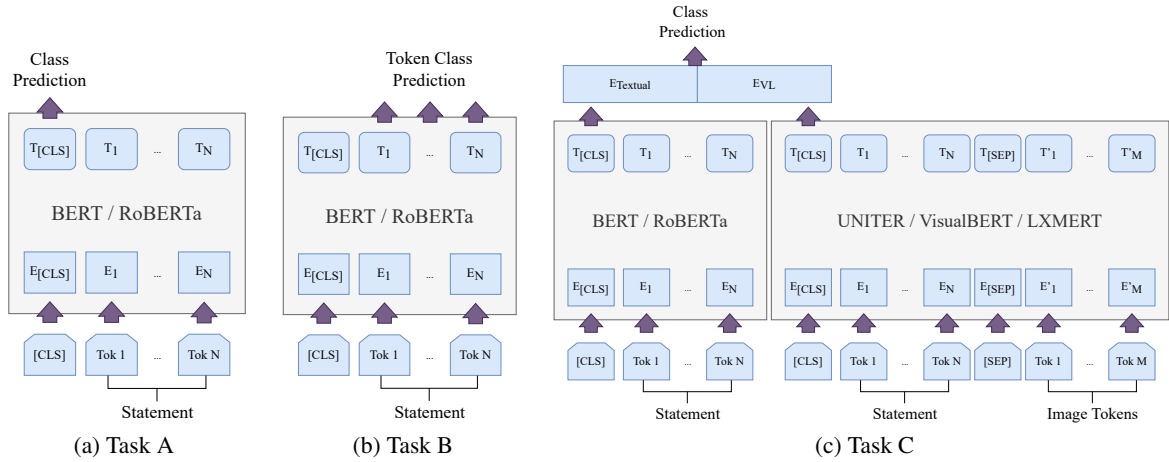


Figure 3: Proposed architectures for the given subtasks

the loss function by using the following equation:

$$\ell(\mathbf{x}, \mathbf{y}) = -\frac{1}{Nd} \sum_{n=1}^N \sum_{k=1}^d [p^k y_n^k \log x_n^k + (1 - y_n^k) \log(1 - x_n^k)]$$

$$p^k = \frac{1}{f^k} (|K| - f^k)$$

(1)

Where  $N$  is the batch size,  $n$  index denotes  $n^{th}$  batch element,  $d$  is the number of classes,  $f$  stands for a vector of class absolute frequencies calculated on the train set,  $\mathbf{x}$  is the output vector from the last Sigmoid layer,  $\mathbf{y}$  is a vector of multi-hot encoded ground truth labels and  $|K|$  is the size of the train set.

We finally use the standard Sigmoid activation function to compute probabilities for each label.

### 3.2 Task B - Span Identification and Text Classification

We experiment with the same BERT-based encoders for this task.

**Pre Processing** Since the spans are given on character-level in the dataset and the transformer models run on token-level, we transform all the spans to token-level by taking the intersection with the tokenized input. Further, we train the model with the obtained token-level labels as targets.

**Model** We model the problem statement as a token-level *multi-label* classification problem for span identification. The model architecture is illustrated in figure 3b. We use the same classifier to classify all token embeddings. To handle class imbalance, we apply weighted Binary Cross-Entropy loss with class weights as mentioned in equation 1

**Post Processing** Since all tokens of a word should belong to the same set of classes, we merge all the tokens of each word and assign a union of those classes to the chosen word.

Finally, all the words are classified using the following equation where  $W_i$  is the set of labels of the  $i^{th}$  word,  $L_{i,j}$  is the set of labels of the  $j^{th}$  token of the  $i^{th}$  word and  $N_i$  is the number of tokens of the  $i^{th}$  word:

$$W_i = \bigcup_{j=1}^{N_i} L_{i,j}$$

We do not apply loss on special tokens so that the classifier is not misled while training.

### 3.3 Task C - Visual-Linguistic Classification

We experiment with Visual-Linguistic (VL) models for this task. Since meme classification is a multi-modal task, a multimodal transformer architecture should be a good choice.

**Pre Processing** The image tokens used in the VL models are extracted using Faster R-CNN (Ren et al., 2015) and a fixed number of image tokens are created by extracting the features of the top 36 regions of interest.

**Model** We first experiment with training the VL models directly. Although the VL models should perform better in the presence of extra cues for classification, poor performance is observed compared to the powerful textual-only models.

On further analyzing the problem, we observe that propaganda detection is a complex semantic problem, and often the classes can be detected by

Parameter	Task A	Task B	Task C	
			VL	Ensemble
Dropout	0.1	-	-	-
Max Sequence Length	128	512	128	128
Batch Size	8	8	16	32
warmup	-	-	0.1	0.1
Learning Rate	1e-05	1e-05	1e-05	1e-05
patience	50	50	50	50
Weight Decay	-	0.01	0.01	0.01
Optimizer	Adam	AdamW	BertAdam	BertAdam

Table 1: Hyperparameters

using just the text. So, we experiment with the textual models and ignore the visual cues in the data. Surprisingly, the textual model outperforms the VL model when trained to learn multimodal labels using just the textual cues. We further study both models to better understand the learnings of each of them (see Section 5) and continue experimentation with ensembling both models.

We propose an ensemble of multimodal transformers like UNITER, VisualBERT, LXMERT with unimodal transformers like BERT and RoBERTa to help the classification model as each architecture can focus on their domains and later merge those embeddings. Rather than using a naive average ensembling method, we propose our own model, illustrated in figure 3c. Our model concatenates the [CLS] token embeddings from transformers in each modality and then applies classification on top of the concatenated vector. The base encoders are fine-tuned unimodally and then frozen while training the ensemble classifier. To train the textual model unimodally, we train the textual model with textual labels only to learn relevant features.

## 4 Experimental Setup

### 4.1 Implementation

We use HuggingFace<sup>1</sup> library (Wolf et al., 2020) to experiment with textual models, and use the official implementations of the multimodal transformers in PyTorch<sup>2</sup> (Paszke et al., 2019). The models were trained with the default hyperparameters with an exception for the parameters mentioned in table 1. All experiments were conducted using Nvidia GeForce RTX 2080 Ti GPU.

For training the models, we use the same train/dev/ test split as mentioned in the task, which is

<sup>1</sup> 🤗 Transformers, v4.2.0, <https://huggingface.co/transformers/>

<sup>2</sup> PyTorch, v1.7.1, <https://pytorch.org/>

in the ratio 10:1:3 with 951 samples in total. We use the best-performing model by comparing the F1-micro scores on the validation set for all our experiments.

### 4.2 Evaluation Metrics

For tasks A and C, the F1-micro score is used as the main performance metric. Due to various low resource classes in the dataset, F1-macro was also used to give weight to the smaller classes in the performance metric, but it was revealed to be highly impacted by small variations in the model.

For task B, we use the official evaluation metric as defined in the task. As the task is a multi-label sequence tagging task, the standard micro-averaged F1 is modified to account for partial matching between the spans.

## 5 Results

We conduct several experiments to compare the performance of models on each of the tasks. Detailed information can be seen in table 2.

For task A, RoBERTa<sub>LARGE</sub> is the best performing model on the task. Although the average performance of the model is worse than BERT<sub>LARGE</sub> on the test set, the maximum performance is still better.

Similarly, for task B, RoBERTa<sub>LARGE</sub> outperforms other models by a large margin.

For task C, we trained the textual models on visual-linguistic labels for a fair comparison with the other VL models. Unexpectedly, they also show decent performance and surprisingly perform better than the multimodal transformers. Finally, the ensemble models are trained with their textual model trained on textual labels and the VL model trained on VL labels. They outperform their unimodal components and end up with a fair performance increase in each of the cases.

During the post-evaluation phase, we conduct more experiments and report the corresponding performances on the test set for the best-validated models on the dev set. We realize that our submitted model performed worse on the test set in task C because the checkpoint used for test set predictions was different from the best validation checkpoint.

The final values we achieve on the test set by using the best validation set checkpoints are 57.56, 48.23, and 55.68 by using RoBERTa<sub>LARGE</sub>, RoBERTa<sub>LARGE</sub> and RoBERTa<sub>LARGE</sub> + UNITER, respectively on all the subtasks.

Model	Task A		Task B		Task C	
	Dev Set	Test Set	Dev Set	Test Set	Dev Set	Test Set
BERT <sub>BASE</sub>	64 ±0.7	52.2 ±2.5	53.1 ±1	45.6 ±1.5	63.3 ±1.2	51.9 ±1
BERT <sub>LARGE</sub>	62.8 ±0.8	<b>54.8</b> ±1.5	53.5 ±2.1	44.7 ±2.5	62.7 ±1.4	52.5 ±0.9
RoBERTa <sub>BASE</sub>	61 ±0.8	51.2 ±1.2	53.2 ±0.6	43.9 ±0.9	61.5 ±1.1	49.8 ±1.2
RoBERTa <sub>LARGE</sub>	<b>64.7</b> ±1.1	53.2 ±3.9	<b>58.5</b> ±2.1	<b>47.6</b> ±1.5	63.9 ±1.1	54.2 ±1
UNITER					64.9 ±0.2	49.2 ±0.6
VisualBERT					57.8 ±0.6	45.8 ±0.7
LXMERT					54.5 ±0.3	44.4 ±0.0
BERT <sub>BASE</sub> + UNITER					66.1 ±0.2	52.9 ±0.8
RoBERTa <sub>LARGE</sub> + UNITER					<b>67.3</b> ±0.9	<b>54.9</b> ±0.6
BERT <sub>BASE</sub> + VisualBERT					63.1 ±0.3	53.1 ±0.5
RoBERTa <sub>LARGE</sub> + VisualBERT					64.4 ±0.5	52.7 ±1.9

Table 2: Mean and std of F1-micro scores computed from 10 runs of the mentioned models. Test set values are reported after choosing the best model for the dev set using early stopping.

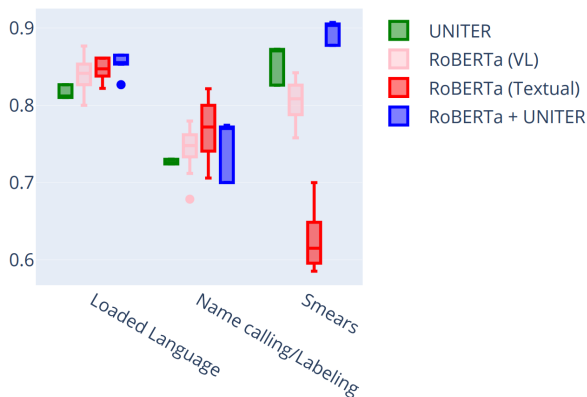


Figure 4: Comparison of different models in the 3 most occurring classes. F1-micro scores of 10 runs are computed on the dev set.

We observe that all BERT-based models give a decent performance for all tasks. An interesting insight we get from the visual-linguistic task is that the problem is not completely multimodal. Several of the persuasive techniques can be identified by using just the textual cues, which is evident from figure 2 as well. While textual models like RoBERTa are pretrained on a much larger textual dataset and are able to learn more complex semantics from the data, VL models suffer when used in textual-only domains. We carry out experiments to further study the differences in these models with different types of labels and propose an architecture that benefits from both models.

A class-wise comparison is carried out for all models. Since several of the classes have very low positive samples, it is difficult to draw conclusions

because of the high variance in performance due to different model initializations, so we compare the most frequently occurring labels only. The difference in the performance of different models is illustrated in figure 4. For this comparison, we train the textual model on multimodal and textual labels to compare the capability of the models. Ideally, the textual models should be trained on textual labels only to learn more relevant features. The ensemble is trained with a textual model trained on textual labels and a VL model trained on VL labels. The comparison shows that the multimodal models are performing much better for detecting *Smears*, which has various samples which require visual cues as well. Another observation is that RoBERTa is performing better on *Loaded Language* and *Name calling / Labelling*. Training an ensemble of the textual model and the multimodal is helping the model perform better in all classes.

To further study the models, we report the performance for each of them in different modalities. To measure the textual performance of the models, we compare the sets of labels in task A and C to shortlist labels which were identified only after the presence of visual cues. We do not consider those predictions and calculate the F1-micro score for the remaining subset of the predictions. Similarly, for measuring the visual performance, we do not consider the labels which were identified by just using the textual cues. The performance of several runs of the models is compared in figure 5. The comparison also supports the claim that ensembling models trained in different modalities help to learn from

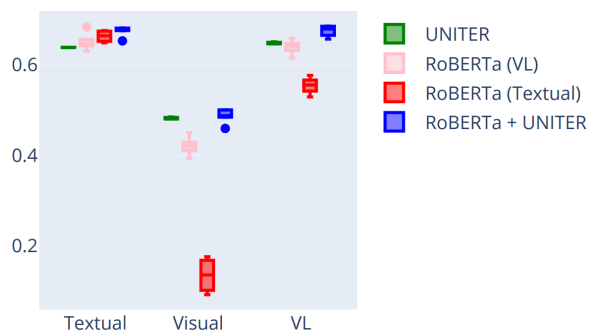


Figure 5: Comparison of different models in different modes. F1-micro scores of 10 runs are computed on the dev set.

the best of both worlds.

## 6 Conclusion

Although detecting persuasive techniques in memes is a multimodal problem, often, most of the techniques can be identified by just using the textual cues from the meme. Since VL models are still in their nascent stages, powerful textual models help with solving the problem at hand. Future work can be done to improve these kinds of problems that are not multimodal in the truest sense. The ensembling method used in our model is very basic; better architectures can be explored to continue this line of work.

## Acknowledgments

We thank the organisers of the shared task for their effort, and the anonymous reviewers for their comments.

## References

- Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.
- Alberto Barrón-Cedeño, Giovanni Da San Martino, Israa Jaradat, and Preslav Nakov. 2019. *Proppy: A system to unmask propaganda in online news*. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9847–9848.
- Akemi Takeoka Chatfield, Christopher G. Reddick, and Uuf Brajawidagda. 2015. *Tweeting propaganda, radicalization and recruitment: Islamic state supporters multi-sided twitter networks*. In *Proceedings of the 16th Annual International Conference on Digital Government Research*, dg.o ’15, page 239–249, New York, NY, USA. Association for Computing Machinery.
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. *Uniter: Universal image-text representation learning*. In *ECCV*.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, and Preslav Nakov. 2019. *Findings of the NLP4IF-2019 shared task on fine-grained propaganda detection*. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 162–170, Hong Kong, China. Association for Computational Linguistics.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. *SemEval-2020 task 11: Detection of propaganda techniques in news articles*. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414, Barcelona (online). International Committee for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. *Task 6 at semeval-2021: Detection of persuasion techniques in texts and images*. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval ’21*, Bangkok, Thailand.
- Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. 2018. *Quantifying controversy on social media*. *Trans. Soc. Comput.*, 1(1).
- Monika Glowacki, Vidya Narayanan, Sam Maynard, Gustavo Hirsch, Bence Kollanyi, Lisa-Maria Neudert, Phil Howard, Thomas Lederer, and Vlad Barash. 2018. *News and political information consumption in Mexico: Mapping the 2018 Mexican presidential election on Twitter and Facebook*. Technical Report COMPROP DATA MEMO 2018.2, Oxford University, Oxford, UK.
- Raul Gomez, Jaume Gibert, Lluís Gomez, and Dimosthenis Karatzas. 2020. *Exploring hate speech detection in multimodal publications*. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*.
- Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and

- Davide Testuggine. 2020. [The hateful memes challenge: Detecting hate speech in multimodal memes](#).
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. Visualbert: A simple and performant baseline for vision and language. In *Arxiv*.
- Clyde R. Miller. 1939. The Techniques of Propaganda. From “How to Detect and Analyze Propaganda,” an address given at Town Hall. The Center for learning.
- Robert Muller. 2018. Indictment of Internet Research Agency. pages 1–37.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Hannah Rashkin, Eunsol Choi, Jin Yea Jang, Svitlana Volkova, and Yejin Choi. 2017. [Truth of varying shades: Analyzing language in fake news and political fact-checking](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937, Copenhagen, Denmark. Association for Computational Linguistics.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. [Faster r-cnn: Towards real-time object detection with region proposal networks](#). In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Shardul Suryawanshi, Bharathi Raja Chakravarthi, Michael Arcan, and Paul Buitelaar. 2020. [Multimodal meme dataset \(MultiOFF\) for identifying offensive content in image and text](#). In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 32–41, Marseille, France. European Language Resources Association (ELRA).
- Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.
- Cristina Tardáguila, Fabrício Benevenuto, and Pablo Ortellado. 2018. Fake News Is Poisoning Brazilian Politics. WhatsApp Can Stop It. <https://www.nytimes.com/2018/10/17/opinion/brazil-election-fake-news-whatsapp.html>.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Ning Xie, Farley Lai, Derek Doran, and Asim Kadav. 2019. Visual entailment: A novel task for fine-grained image understanding. *arXiv preprint arXiv:1901.06706*.
- Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

# MinD at SemEval-2021 Task 6: Propaganda Detection using Transfer Learning and Multimodal Fusion

Junfeng Tian, Min Gui, Chenliang Li, Ming Yan, Wenming Xiao

Alibaba Group, China

{tjf141457, guimin.gm, lcl193798, ym119608, wenming.xiaowm}@alibaba-inc.com

## Abstract

We describe our systems of subtask1 and subtask3 for SemEval-2021 Task 6 on *Detection of Persuasion Techniques in Texts and Images*. The purpose of subtask1 is to identify propaganda techniques given textual content, and the goal of subtask3 is to detect them given both textual and visual content. For subtask1, we investigate transfer learning based on pre-trained language models (PLMs) such as BERT, RoBERTa to solve data sparsity problems. For subtask3, we extract heterogeneous visual representations (i.e., face features, OCR features, and multimodal representations) and explore various multimodal fusion strategies to combine the textual and visual representations. The official evaluation shows our ensemble model ranks 1st for subtask1 and 2nd for subtask3.

## 1 Introduction

With the recent interest in “fake news”, the detection of propaganda or highly biased texts has emerged as an active research area (Da San Martino et al., 2020, 2019; Chernyavskiy et al., 2020).

SemEval-2021 Task 6 (Dimitrov et al., 2021) provides three subtasks aiming to detect persuasion techniques in texts and images. We participate in subtask1 and subtask3, which are defined as follows:

- **subtask1:** Given only the “textual content” of a meme, identify which of the 20 techniques are used in it. This is a multilabel classification problem.
- **subtask3:** Given a meme, identify which of the 22 techniques are used both in the textual and visual content of the meme (multimodal task).

For subtask1, we focus on using transfer learning to tackle problems related to the scarcity of data

since deep learning models require a whole lot of data while it is difficult to obtain vast amount of the labeled data. Especially, we first fine-tune the pre-trained language models on an external dataset from SemEval-2020 Task 11 (Da San Martino et al., 2020) and then continue to fine-tune them on the training dataset of SemEval-2021 Task 6. The probabilities of these tuned models are averaged to make the final prediction.

For subtask3, we concentrate on multimodal fusion to combine textual and visual representation. Heterogeneous visual representations are extracted, including face, OCR and multimodal representations. Face representation consists of recognized human faces and facial expressions. OCR representation can capture the relations among snippets in an image. Multimodal pre-trained model is capable of simultaneously processing multimodality inputs for joint visual and textual understanding. After that, we explore three multimodal fusion strategies (i.e., Average, Concat and MLP) to combine the textual and visual representations.

The experimental results show that transfer learning can leverage knowledge from source data to tackle problems related to the scarcity of data, and heterogeneous visual representation (i.e., face, OCR, and multimodal representation) can be used as complementary features to better detect persuasion techniques. Our ensemble model ranks 1st for subtask1 and 2nd for subtask3.

## 2 System Overview

In this section, we provide a general overview of our systems for the two subtasks. We consider the propaganda detection task as multimodal multi-class multi-label classification task, predicting one or more labels given an input text and an input image.

## 2.1 Model

Various pre-trained models are explored to extract textual and visual features, and these textual and visual features are fused to predict labels.

**Textual Representation** In this paper, five pre-trained language models (PLMs) are used. Representations of the special token [CLS] are passed to the classification layer. We briefly describe each PLM:

- **BERT** (Devlin et al., 2019) is a powerful transformer-based PLM and enables bidirectional training using a “masked language model” (MLM) pre-training objective. The masked language model randomly masks some input tokens and aims to predict the masked tokens. BERT also use next sentence prediction (NSP) objective during pretraining, which is a binary classification loss for predicting whether two segments follow each other in the original text. With tailored finetune objectives, BERT can improve performance on downstream tasks such as classification tasks.
- **RoBERTa** (Liu et al., 2019) proposes an improved recipe for training BERT models and boosts the performance on GLUE(Wang et al., 2019), RACE(Lai et al., 2017) and SQuAD(Rajpurkar et al., 2016). It shares the same model architecture with BERT, and mainly improves BERT by dynamic masking and a larger byte-level Byte-Pair Encoding (BPE)(Sennrich et al., 2016).
- **XLNet** (Yang et al., 2019) integrates the segment recurrence mechanism and relative encoding scheme of Transformer-XL(Dai et al., 2019) into pretraining with reparameterizing. It can capture the dependency between the masked positions and alleviate a pretrain-finetune discrepancy.
- **DeBERTa** (He et al., 2020) disentangles attention mechanism and encodes each word with two vectors representing content and position, respectively. An enhanced mask decoder is also used to incorporate absolute positions in the decoding layer to predict the masked tokens in model pre-training. These methods enables DeBERTa to obtain competitive performance of both natural language understand (NLU) and natural language generation (NLG) downstream tasks.
- **ALBERT** (Lan et al., 2019) replaces the next sentence prediction (NSP) loss with a sentence order prediction (SOP) loss to better model inter-sentence coherence. Besides, it equips two parameter reduction techniques to lower memory consumption and increase the training speed of BERT. With fewer parameters compared to BERT-large, ALBERT establishes new state-of-the-art results on the GLUE, RACE, and SQuAD benchmarks.

**Visual Representation** Three visual representations are adopted, including face representation, OCR representation and single-stream multimodal representation.

- **Face Representation** It is important to recognize faces and facial expressions for propaganda detection. We use a state-of-the-art *face recognition* model, which is a ResNet-34 network (He et al., 2016) with 29 conv layers. In an image, each face is encoded as a 128 dimensional vector using the published toolkit<sup>1</sup> and adopt mean pooling for the final face representation.
- **OCR Representation** For text in an image, the 2-D position of the text can capture the font size and the relationship among tokens within the image. Therefore, we use a 2-D position embedding to jointly model interactions between text and layout information across the image. We extract the bounding box 2-D position using the Microsoft OCR<sup>2</sup>.
- **Multimodal Representation** Recent studies on vision-language pre-training have pushed the limits of a variety of Vision-and-Language (V+L) tasks, and both the image and text content can help understand the semantics of the meme for propaganda detection. Therefore, we also extract a region-based image features with Faster R-CNN (Ren et al., 2015) to represent the image. Then, we follow (Li et al., 2021) and use a pre-trained multi-modality model SemVLP to better learn the multimodal fusion between the image and text.

**Multimodal Fusion** For multimodal propaganda detection, we employ 3 fusing methods to combine the textual and visual features.

<sup>1</sup>[https://github.com/ageitgey/face\\_recognition](https://github.com/ageitgey/face_recognition)

<sup>2</sup><https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/concept-recognizing-text>

- **Average** The predicted probabilities of text and image features are averaged for prediction:

$$\hat{y}_c = (\text{sigmoid}(W_1(\tanh(W^t h^t) + b_1)) + \text{sigmoid}(W_2(\tanh(W^v h^v) + b_2)))/2$$

where  $h_t$  and  $h_v$  stand for textual and visual representations, respectively.

- **Concat** The text and image features is concatenated to predict probabilities:

$$\hat{y}_c = \text{sigmoid}(W[h^t, h^v] + b)$$

- **MLP** Before making prediction, we map text and image features to the same semantic space:

$$\hat{y}_c = \text{sigmoid}(W(\tanh(W^t h^t + W^v h^v) + b))$$

## 2.2 Training

**Multilabel Classifier** We provide an additional label-wise feed-forward network (FFN) and a linear layer to extract label. At training time, we propose to minimize the binary cross-entropy (BCE) objective  $\mathcal{L}$  as follows:  $\mathcal{L}_{\text{BCE}}(\hat{y}_c, y_c) = -y_c \log \hat{y}_c - (1 - y_c) \log (1 - \hat{y}_c)$  where  $y_c$  is the ground truth of class  $c$  and  $\hat{y}_c$  is the predicted value. At test time, we predict the label as  $\tilde{y}_c = \mathbb{I}(\hat{y}_c > T)$  where  $T$  is a probability threshold and  $\mathbb{I}$  is the indicator function.

As for label imbalance problem, focal loss (FL) (Lin et al., 2017), which down-weights easy examples and focus training on hard negatives, is adopted during training.

**Transfer Learning** It is difficult to get vast amounts of labeled data for supervised models. Transfer Learning enables us to utilize knowledge from previously learned tasks and apply them to newer, related ones. We use transfer learning from the news articles domain: we first train the model using the news data, and then we continue training for this task. In preliminary experiments, we find that fine-tuning layers in the process is better than freezing them as feature extractors.

## 3 Experimental Setup

### 3.1 Dataset

We conduct experiments with the train, the dev and the test datasets provided by SemEval-2021 Task 6 (Dimitrov et al., 2021), which contains 687, 63 and 200 memes for subtask1 and subtask3, respectively.

**External Resources** We use the annotations of the PTC corpus (more than 20,000 sentences) from SemEval-2020 task 11 (Da San Martino et al., 2020) as external resource. Although its domain is news articles and fewer techniques are considered, the annotations are made using the same guidelines as SemEval-2021 task 6.

### 3.2 Evaluation Measures

Subtask1 and subtask3 are multi-label classification tasks. The official evaluation measure for both tasks is micro-F1. We also report macro-F1.

### 3.3 Parameter Settings

We adopt the large models and select hyper-parameters using validation on a subsample of the training data. The cased models are used because that upper cases contain strong emotion signals in this task. We use adamW optimizer (Loshchilov and Hutter, 2019) with 500 warm-up steps and train for 10 epochs with a  $2e-5$  learning rate and a 8 batch size. The last checkpoint is used for evaluation.

### 3.4 Submitted Systems

**Post-processing** *Repetition* means repeating the same message over and over again so that the audience will eventually accept it. Therefore, we assign a *Repetition* label in case if there exists a bigram appears more than 3 times.

**Ensemble** We use model ensemble for final submission. In particular, for subtask1 we explore 5 pre-trained models (using BCE Loss, Focal Loss and Transfer Learning, respectively), and for subtask3 we additionally explore face, OCR, multimodal representations and the fusion strategies. We take the probabilities of these settings and average them to make the final prediction.

### 3.5 Test Results

Table 1 and Table 2 list the results of the top-performing teams for subtask1 and subtask3. We can see that our proposed model is ranked 1st for subtask1 and 2nd for subtask3 among all teams.

## 4 Discussion

More thorough studies and analyses are conducted in this section, trying to answer two questions: (1) How is the performance of transfer learning on less data? (2) How is the performance of multimodal fusion on multimodal data? Moreover, we give



Rank	Team	F1-Macro	F1-Micro
1	<b>MinD</b>	<b>0.28993</b>	<b>0.59331</b>
2	Alpha	0.26218	0.57187
3	Volta	0.26621	0.56958
	Baseline	0.04427	0.06439

Table 1: Results of top 3 teams for **subtask1 (test)**.

Rank	Team	F1-Macro	F1-Micro
1	Alpha	<b>0.27315</b>	<b>0.58109</b>
2	<b>MinD</b>	0.24389	0.56623
3	1213Li	0.22830	0.54860
	Baseline	0.05152	0.07062

Table 2: Results of top 3 teams for **subtask3 (test)**.

error analyses on the test dataset to provide an overview of problematic labels.

#### 4.1 Transfer Learning

We perform ablation study for each PLM (row) and each learning method (column) in Table 3 for subtask1. It shows that:

First, RoBERTa and DeBERTa were generally the best performing models. Given that RoBERTa and DeBERTa are carefully tuned models base on BERT, this result is reasonable.

Second, both Focal Loss and Transfer Learning help to alleviate data sparsity problems. Focal Loss help DeBERTa and ALBERT improve 0.7 and 0.6 points. Because Focal Loss assigns higher weights to sparse samples and reduces the weights to frequent samples. Transfer Learning helps BERT, RoBERTa, XLNet improve 1.0, 4.0, 5.7 points, respectively. RoBERTa with Transfer Learning achieves the best single model score. Transfer Learning help transfer the parameters trained on related data or task to the newer model. Instead of learning from scratch, the newer model can leverage knowledge to tackle problems related to the scarcity of data.

#### 4.2 Multimodal Fusion

For subtask3, we compare different multimodal representations in Table 4 and fusion strategies in Table 5. We find that:

(1) both OCR Representation and Multimodal Representation models outperform the Text Representation model. OCR Representation can additionally capture the relative space relationship instead of sequential information among texts in an image.

Training PLM	BCE	FL	Transfer
BERT	0.5833	0.5552	<b>0.5941</b>
RoBERTa	0.6070	0.5950	<b>0.6478</b>
XLNet	0.5573	0.5418	<b>0.6148</b>
DeBERTa	0.6307	<b>0.6378</b>	0.6230
ALBERT	0.5251	0.5319	<b>0.5081</b>

Table 3: Results (**F1-Micro**) for **subtask1 (dev)**. **BCE**, **FL**, **Transfer** stand for models training using BCE Loss, Focal Loss and Transfer Learning, respectively.

Model	F1-Macro	F1-Micro
Text Representation	0.2481	0.5012
Face Representation	0.1956	0.2332
OCR Representation	<b>0.2722</b>	0.5208
Multimodal Representation	0.2355	<b>0.5876</b>

Table 4: Results for **subtask3 (dev)**. We explore various multimodal representations.

(2) Multimodal Representation model achieves the best single model performance since it jointly aligns the semantics between image and text and thus is effective for the vision-language understanding task.

(3) Table 5 lists the results of different fusion strategy. We combine the text and face representations since they are the minimal semantic elements in the image. *Concat* obtains the best result on both Macro-F1 and Micro-F1 metrics, though it is the simplest strategy for fusion.

#### 4.3 Error Analysis

To provide an overview of problematic labels, We give error analysis in Table 6 and Table 7. We find that: (1) *Loaded Language and Name Calling*, which are the most frequent labels, show reasonably good performance (0.8190 and 0.6667 F1 score).

(2) On the other hand, as to labels with fewer training samples (less than 20), the system tends not to predict. Additionally, we find rules for *Repetition* do not work and all the predicted label are wrongly classified.

(3) *Slogans, Glittering generalities and Smears* are relative hard to identify. Meanwhile, Recall values of *Transfer and Strong Emotions* for subtask3 are less than 0.1. It lacks enough training samples to well fit the network parameters.

Fusion	F1-Macro	F1-Micro
Average	0.3673	<b>0.6114</b>
MLP	0.3947	0.6094
Concat	<b>0.4218</b>	<b>0.6114</b>

Table 5: Results for **subtask3 (dev)**. We explore various multimodal representations.

Label	Precision	Recall	F1	#
Appeal to authority	-	-	-	13
Appeal to fear	0.4615	0.6000	0.5217	43
B&W	0.6667	0.2857	0.4000	18
Oversimplification	0.4000	0.6667	0.5000	27
Doubt	0.5294	0.3214	0.4000	48
Exaggeration	0.5238	0.5789	0.5500	52
Flag-waving	0.5714	0.6667	0.6154	27
Glittering generalities	0.6667	0.1818	0.2857	32
Loaded Language	0.7197	0.9500	0.8190	358
Straw Man	-	-	-	20
Name calling	0.5658	0.8113	0.6667	218
Obfuscation	-	-	-	4
Red Herring	-	-	-	1
Reductio ad hitlerum	-	-	-	9
Repetition	-	-	-	8
Slogans	0.2857	0.1053	0.1538	44
Smears	0.3864	0.7556	0.5113	200
Cliché	-	-	-	20
Whataboutism	0.5000	0.3000	0.3750	40
Bandwagon	-	-	-	2

Table 6: Precision, Recall and F1 of each label for **subtask1 (test)**. The last column (#) stands for the number of training samples.

## 5 Conclusion

In this paper, we adopt transfer learning to handle data sparsity problems for subtask1, and fuse heterogeneous multimodal representation for subtask3. The experimental results show that transfer learning can leverage knowledge from source data to tackle problems related to the scarcity of data, and heterogeneous visual representation (i.e., face, OCR, and multimodal representation) can extract complementary features.

In future work, we plan to explore fine-grained multimodal fusion with token representations in text and object features in images.

## References

- Anton Chernyavskiy, Dmitry Ilvovsky, and Preslav Nakov. 2020. Aschern at semeval-2020 task 11: It takes three to tango: Roberta, crf, and transfer learning. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1462–1468.
- Giovanni Da San Martino, Alberto Barrón-Cedeno, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. Semeval-2020 task 11: Detection of

Label	Precision	Recall	F1	#
Appeal to authority	-	-	-	19
Appeal to fear	0.6667	0.2222	0.3333	66
B&W	1.0000	0.2857	0.4444	19
Oversimplification	0.3333	0.7500	0.4615	31
Doubt	0.6364	0.1707	0.2692	61
Exaggeration	0.6667	0.3871	0.4898	60
Flag-waving	0.5556	0.4167	0.4762	36
Glittering generalities	0.2857	0.0833	0.1290	84
Loaded Language	0.7333	0.8800	0.8000	360
Straw Man	-	-	-	32
Name calling	0.6329	0.7692	0.6944	252
Obfuscation	-	-	-	5
Red Herring	-	-	-	2
Reductio ad hitlerum	-	-	-	15
Repetition	-	-	-	10
Slogans	0.2857	0.0909	0.1379	45
Smears	0.5348	0.9524	0.6849	450
Cliché	-	-	-	20
Whataboutism	1.0000	0.1429	0.2500	47
Bandwagon	-	-	-	2
Transfer	0.6667	0.0870	0.1538	61
Strong Emotions	1.0000	0.0526	0.1000	68

Table 7: Precision, Recall and F1 of each label for **subtask3 (test)**. The last column (#) stands for the number of training samples.

propaganda techniques in news articles. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1377–1414.

Giovanni Da San Martino, Seunghak Yu, Alberto Barrón-Cedeño, Rostislav Petrov, and Preslav Nakov. 2019. Fine-grained analysis of propaganda in news articles. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, Hong Kong, China.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Dimiter Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at semeval-2021: Detection of persuasion techniques in texts and images. In *Proceedings of the*

- 15th International Workshop on Semantic Evaluation, SemEval '21, Bangkok, Thailand.*
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.
- Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017. RACE: large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9–11, 2017*, pages 785–794. Association for Computational Linguistics.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Chenliang Li, Ming Yan, Haiyang Xu, Fuli Luo, Wei Wang, Bin Bi, and Songfang Huang. 2021. Sem{vlp}: Vision-language pre-training by aligning semantics at multiple levels.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.
- Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7–12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019*. OpenReview.net.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

# CSECU-DSG at SemEval-2021 Task 6: Orchestrating Multimodal Neural Architectures for Identifying Persuasion Techniques in Texts and Images

Tashin Hossain, Jannatun Naim, Fareen Tasneem,  
Radiathun Tasnia, and Abu Nowshed Chy

Department of Computer Science and Engineering  
University of Chittagong, Chattogram-4331, Bangladesh

{tashin.hossain.cu, jannatun.naim.cu, fareen.tasneem,  
radia.tasnia.cu}@gmail.com and nowshed@cu.ac.bd

## Abstract

Inscribing persuasion techniques in memes is the most impactful way to influence peoples' mindsets. People are more inclined to memes as they are more stimulating and convincing and hence memes are often exploited by tactfully engraving propaganda in its context with the intent of attaining specific agenda. This paper describes our participation in the three subtasks featured by SemEval 2021 task 6 on the detection of persuasion techniques in texts and images. We utilize a fusion of logistic regression, decision tree, and fine-tuned DistilBERT for tackling subtask 1. As for subtask 2, we propose a system that consolidates a span identification model and a multi-label classification model based on pre-trained BERT. We address the multi-modal multi-label classification of memes defined in subtask 3 by utilizing a ResNet50 based image model, DistilBERT based text model, and a multi-modal architecture based on multikernel CNN+LSTM and MLP model. The outcomes illustrated the competitive performance of our systems.

*Keywords:* persuasion techniques, transfer learning, multimodal neural architecture.

## 1 Introduction

Persuasion techniques are quite recurrent in social media contents as it reaches a vast community. Proselytizing contents are adroitly implanted in posts and blogs which influence people's thoughts unconsciously. Nowadays such techniques are also being instilled in memes as people's attention is easily captured through illustration rather than narration. Manipulators often use this as a tool to promote their own deceitful agenda which can be political or anything else. Fake news is also spread through these disguised duplicitous contents which

cause a lot of casualties. Therefore, it is an indispensable task to detect these techniques in multimodal contents to protect the users from deception.

The objective of SemEval 2021 task 6 (Dimitrov et al., 2021) is to detect the persuasion techniques in textual and multi-modal contents. This task includes three subtasks where the first two are based on textual contents only. More precisely, the first subtask requires us to detect which persuasion techniques among the given 20 techniques are inscribed in the textual content whereas the second subtask requires us to not only find which techniques are used but also to find the specific span of the text each technique corresponds to. The third subtask is a multi-modal multi-label classification problem where we need to identify which of the given 22 techniques are engraved both in the textual and visual content of the meme. An example from the provided dataset along with the desired output for three subtasks is depicted in Figure 1.

Numerous works have been done on the multi-label classification of text contents. (Chalkidis et al., 2019) depicted the pre-eminent impact of bidirectional GRU with label-wise attention in the legal domain. A consolidation of latent emotion memory (LEM) network and Bi-GRU was exploited for multilabel emotion classification (Fei et al., 2020). Besides, SemEval 2020 task 11 (Da San Martino et al., 2020) introduced two subtasks including span identification of propagandistic fragments in text content and technique classification of propagandistic fragments. The top-performing team (Morio et al., 2020) in the span identification subtask utilized several pre-trained language models for both subtasks. They also proposed an effective ensemble method with stacked generalization. The winning team (Jurkiewicz et al., 2020) of the technique classification subtask approached with an ensemble of RoBERTa based models and utilized RoBERTa-CRF archi-

---

The first four authors have equal contributions.

Tasks	Input	Output (Persuasion Techniques)
Subtask #1	ELEGANT AT LYING BRUTAL WITH THE TRUTH	<ul style="list-style-type: none"> <li>Loaded Language</li> <li>Exaggeration / Minimisation</li> </ul>
Subtask #2		<i>BRUTAL</i> : Loaded Language <i>BRUTAL WITH THE TRUTH</i> : Exaggeration / Minimisation
Subtask #3	ELEGANT AT LYING BRUTAL WITH THE TRUTH + Meme:113_image.png	<ul style="list-style-type: none"> <li>Exaggeration / Minimisation</li> <li>Glittering Generalities (Virtue)</li> <li>Loaded Language</li> <li>Smears</li> </ul>

Figure 1: An illustration of the different subtasks.

texture for the span identification subtask. (Wen et al., 2020) addressed a multi-label image classification problem by following human behavior pattern where labels and image features extracted by the ConvNet were projected to a common latent vector space to capture label correlation. (Song et al., 2018) used a deep multi-modal CNN method for multi-instance multi-label image classification.

In this paper, we present our approaches to address the challenges of identifying persuasion techniques in the textual and multimodal contents as defined in SemEval 2021 task 6. We exploit various kinds of approaches ranging from traditional statistical classifiers to the state-of-the-art deep learning architecture (e.g. multi-kernel CNN+LSTM, MLP, and ResNet50) and transformer models (e.g. BERT, DistilBERT, and FastBERT) in our proposed unified architecture.

We arrange the rest of the paper as follows: we explicate our proposed framework in Section 2. Section 3 unfolds the experimental details and comparative performance analysis. We analyze the performance of our models and also portray an analysis of erroneous detection in Section 3.4. Finally, we conclude this paper with some future prospects in Section 4.

## 2 Proposed Architecture

### 2.1 Subtask 1: Multi-label Persuasion Techniques Classification

In subtask 1, we need to design a method to identify the persuasive techniques used in textual content of a meme. The overview of our proposed system is depicted in Figure 2. In our proposed system, we combine three different models: 1) Logistic regression classifier, 2) Decision tree classifier, and 3)

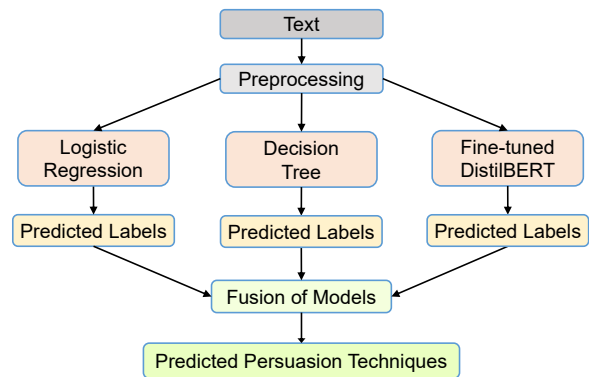


Figure 2: Proposed framework of Subtask 1.

Fine-tuned DistilBERT model. We apply some pre-processing techniques including removing punctuations, numbers, special and single characters, multi-space, text lower-casing, word contradiction, and lemmatizing (Loper and Bird, 2002). Using our proposed models, we get different probability values for corresponding labels. Comparing our threshold score against the probability values, we find multi-label predictions from the individual models and employ the majority voting scheme to obtain our final multi-label predictions.

#### 2.1.1 Logistic Regression

Logistic regression (Cheng and Hüllermeier, 2009) is a machine learning model using probability concepts. It exploits some set of discrete values and the result is converted into a probability score by using a logistic sigmoid function. In our system, we employ a Tf-Idf vectorizer scheme for effective feature representation. We fix our threshold score to 0.05 for converting the probability score into a specific label category. If the probability score is greater than threshold values, it returns 1 as a true value for a specific label and vice versa.

### 2.1.2 Decision Tree

The decision tree (Safavian and Landgrebe, 1991) is a supervised learning classifier where values are divided ceaselessly following some specific parameters. We divide the decision tree into two sub-components, one is decision nodes which split our values and another is leaves which are considered as final decided outcomes. For multi-label classification, we get different probabilities for all the class labels and set the threshold value to select labels following the same process as employed in the logistic regression.

### 2.1.3 Fine-tuned DistilBERT

DistilBERT (Sanh et al., 2019) is a transformer model that has 40% fewer parameters than BERT-base and works 60% faster. We fine-tuned DistilBERT model using the training dataset. For training purposes, we format the pre-processed data into two columns. One column contains the pre-processed text, and the other column carries labels. We convert the labels using scikit-learn (Pedregosa et al., 2011) MultiLabelBinarizer. We construct a neural network named DistilBERTClass involving the DistilBERT model along with the dropout and linear layer on top of it. The dimension of the linear layer is 20 which is the number of labels given in our subtask. We train the model a couple of times by feeding our dataset and we get the probability of each label. We use a random threshold to select the final labels.

### 2.1.4 Fusion of Models

We assemble our three individual models through a majority voting scheme. In majority voting, we count the occurrences of labels from three distinct models. We append the labels with the frequency of 2 or more to the final list of labels. Therefore, we obtain our final list of persuasive techniques for a given meme text.

## 2.2 Subtask 2: Span Identification of Persuasive Techniques

We propose a system that integrates a span identification model and a multi-label classification model for this subtask. We exploit an approach based on pre-trained BERT (bert-base-uncased). We employ SemEval 2020 Task 11’s (Da San Martino et al., 2020) propaganda dataset as an external corpus. The overview of our proposed model is depicted in Figure 3.

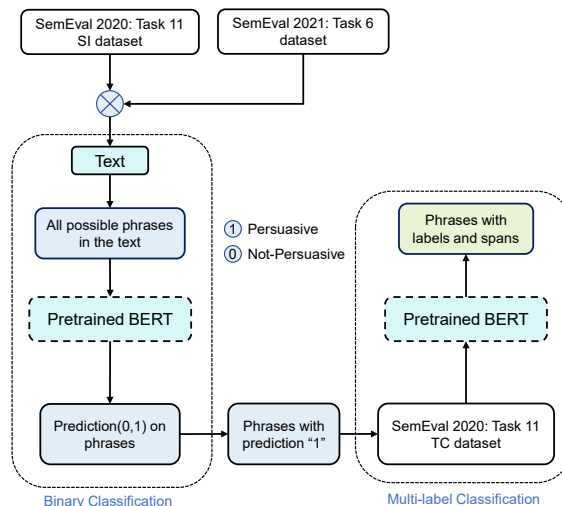


Figure 3: Proposed framework of Subtask 2.

### 2.2.1 Span Identification

We accumulate the sentences extracted from the articles of SemEval 2020 Task 11’s SI dataset, SemEval 2021 Task 6’s train, and development dataset. We derive all possible phrases from these sentences. Phrases with their indices included in span are labeled as 1 (Persuasive) while others are labeled as 0 (Not persuasive). This customized dataset is then sent to the pre-trained BERT model (Devlin et al., 2019) for training. We also extract all possible phrases from the test dataset. The pre-trained BERT model conducts binary classification on this test set. Here, the phrases are considered as sentences, so this process can be comprehended as a binary sentence classification task. After classifying the phrases derived from the test dataset, the indices of the phrases classified as 1 (Persuasive) are included in the spans and further processed for technique classification.

### 2.2.2 Technique Classification

The phrases of the test data that are predicted as persuasive in the previous segment are used as the test dataset of this segment. In this portion, we congregate SemEval 2020 Task 11’s technique classification dataset, SemEval 2021 Task 6’s train, and development dataset. In the case of the last two of them, we only include the text fragments, the indices of which are included in the provided spans instead of the whole text. We then send this contrived trainset to another pretrained BERT model with the same configuration as before and operate multi-label classification on the test set which generates predicted labels among the given 20 labels

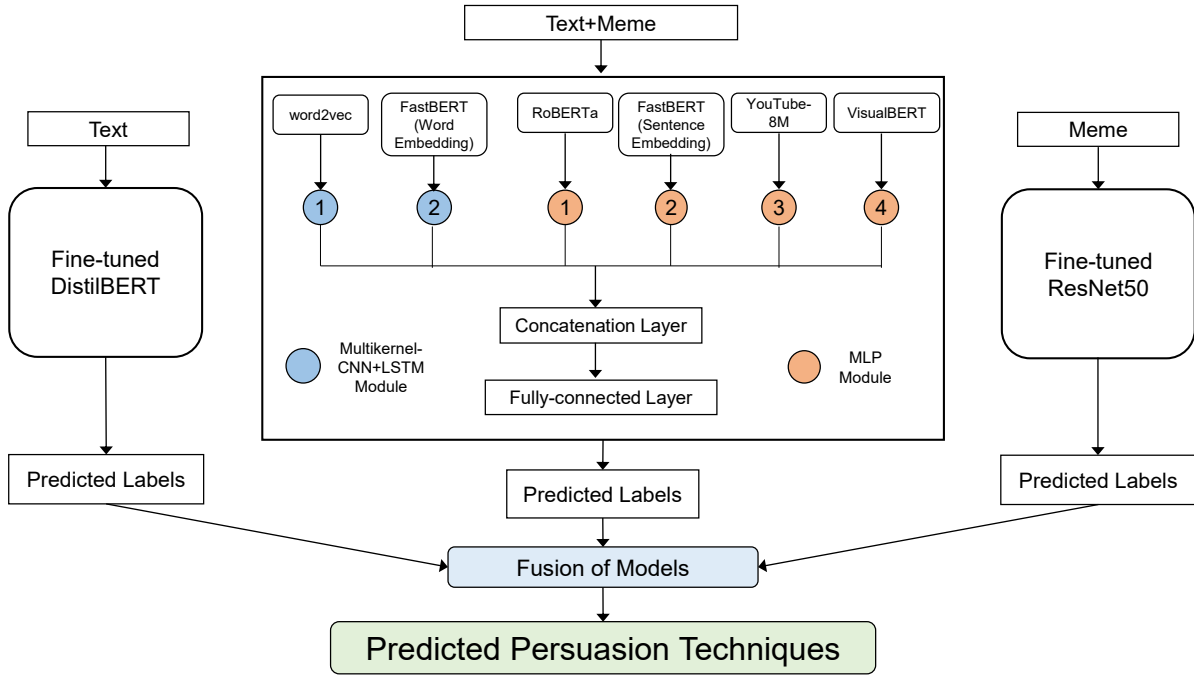


Figure 4: Proposed framework of Subtask 3.

per phrase. The phrases, their start index, end index, and their corresponding labels are then reintegrated as text fragments, start index, end index, and technique accordingly with their original text and converted into a suitable format for submission.

### 2.3 Subtask 3: Multi-modal Multi-label Classification

For this multi-modal subtask, we propose a majority voting based architecture as illustrated in Figure 4. We exploit a fine-tuned DistilBERT model, an ensemble of multi-kernel CNN with LSTM module and MLP module, and a fine-tuned ResNet50 model. These three models produce a list of persuasive techniques singularly and these lists are passed to the majority voting module to obtain the final list of persuasive techniques.

#### 2.3.1 Fine-tuned DistilBERT

We use the same process of training described in Section 2.1.3. We accumulate the training and development dataset in a single corpus. Later, we use the 90% percent of the data for training and the rest of used as the validation set for finetuning.

#### 2.3.2 Fine-tuned ResNet50

We perform fine-tuning on the residual neural network (He et al., 2016) having 50 layers. We convert our meme dataset as the format of the iMet Collection 2019 - FGVC6 dataset (Zhang et al., 2019).

For training purposes, we include an additional label for the memes which have no labels assigned. We utilize the “ResNet50” pre-trained model, having “imagenet” as weights and 1000 classes. We interchange the Average pool layer with the AdaptiveAvgPool2d layer. We attach some batch normalization layers, dropout, and a linear layer. In the linear layer, the BatchNorm1d takes 2048 features as input. In the output layer, we return 23 output features where we add one additional label with the number of labels given in our problem. We train two layers such as layer4 and the last linear layer with the corresponding learning rate  $1e-5$  and  $5e-3$ . We train the model numerous times and then get the model predictions. Finally, we set a random threshold to get the final predicted labels.

#### 2.3.3 Ensemble of Multi-kernel CNN + LSTM and MLP Model

To address the challenge of the multimodal subtask, a combination of high-level features in a neural architecture is conventional. Our proposed model suggests a fusion of features extracted from multi-kernel CNN on top of the LSTM model and MLP (multi-layer perceptron) model. We exploit two kinds of word embeddings including word2vec (Mikolov et al., 2013) and fine-tuned FastBERT (Liu et al., 2020) models which are sent to the convolutional model of kernel size (2, 3) and subsequently to the LSTM model.

Besides, we also explore a multi-layer perceptron model for one-dimensional image features, sentence embeddings, and multi-modal features.

- **Image Features:** The image features are extracted from YouTube-8M (Abu-El-Haija et al., 2016) image feature extractor model(1024-dimension).
- **Sentence Embeddings:** These are extracted from the fine-tuned FastBERT (768-dimension) model and pre-trained RoBERTa (768-dimension) (Liu et al., 2019) model.
- **Multi-modal Features:** VisualBERT (Li et al., 2019) is exploited to blend image features along with text features. We implement the model proposed by (Li et al., 2020). We extract the image features utilizing Detectron2 (Wu et al., 2019) and the text features are encoded from a pre-trained BERT model. Both features are then merged inside VisualBERT. The dimension of the features is (164,768) and we flatten these features for our MLP module.

The output from two multi-kernel CNN+LSTM (MKCNN+LSTM) modules and four MLP modules are concatenated and further transmitted to the fully connected layer.

### 2.3.4 Fusion of Models

The list of predicted labels from the above three models are subsequently passed to a majority voting module. The primary idea behind majority voting is based on the frequency of the labels. If a label exists in the majority of the models, it is appended in the final list of labels.

## 3 Experiments and Evaluations

### 3.1 Dataset Description

In SemEval-2021 task 6 (Dimitrov et al., 2021), overall 950 data has been provided for subtask 1, 2, and 3. In the case of subtask 1 and 2, training, development, and test set contain 687, 63, and 200 textual data respectively. For subtask 3, the same amount of textual and meme data has been accommodated since it is a multi-modal subtask. Dataset for subtask 1 and 3 is annotated with 20 and 22 persuasive techniques correspondingly while subtask 2 dataset provides spans of 20 techniques used all together in the text.

### 3.2 Experimental Setup

In this section, we illustrate our submitted systems in SemEval-2021 Task 6. In case of subtask 1, we use three different models i.e. logistic regression, decision tree classifier, and fine-tuned DistilBERT model. The system configuration of these three individual models are given in Table 1.

System	Settings
Logistic Regression	<ol style="list-style-type: none"> <li>1. <i>max_iter</i>: 2000</li> <li>2. <i>C</i>: 20</li> <li>3. <i>penalty</i>: l2</li> <li>4. <i>tol</i>: 0.001</li> </ol>
Decision Tree	<ol style="list-style-type: none"> <li>1. <i>min_samples_split</i>: 2</li> <li>2. <i>min_samples_leaf</i>: 1</li> <li>3. <i>criterion</i>: gini</li> <li>4. <i>splitter</i>: best</li> </ol>
Fine-tuned DistilBERT	<ol style="list-style-type: none"> <li>1. <i>Tokenizer</i>: distilbert-base-uncased</li> <li>2. <i>Dropout</i>: 0.2</li> <li>3. <i>Learning rate</i>: 2e-5</li> <li>4. <i>Batch size</i>: 16</li> <li>5. <i>num_workers</i>: 4</li> <li>6. <i>Maximum length</i>: 60</li> <li>7. <i>Epochs</i>: 10</li> </ol>

Table 1: System settings for Subtask 1.

We used the same system configuration of pre-trained BERT model for two segments i.e. span identification and multi-label technique classification in the subtask 2. The system settings are depicted in Table 2.

System	Settings
Pre-trained BERT	<ol style="list-style-type: none"> <li>1. <i>max_seq_length</i>: 128</li> <li>2. <i>Epochs</i>: 1</li> <li>3. <i>train_batch_size</i>: 8</li> <li>4. <i>eval_batch_size</i>: 8</li> <li>5. <i>Weight decay</i>: 0.5</li> <li>6. <i>Learning rate</i>: 4e-5</li> <li>7. <i>adam_epsilon</i>: 1e-8</li> <li>8. <i>warmup_ratio</i>: 0.06</li> <li>9. <i>max_grad_norm</i>: 1.0</li> <li>10. <i>gradient_accumulation_steps</i>: 1</li> <li>11. <i>logging_steps</i>: 50</li> <li>12. <i>save_steps</i>: 2000</li> </ol>

Table 2: System settings for Subtask 2.



For subtask 3, we used three types of models. One is a fine-tuned DistilBERT model which is trained using the text written in a meme. The other model is a fine-tuned ResNet50 model, and the last one is multi-kernel CNN+LSTM and MLP model. These three models trained with the given dataset using different parameter settings. The system settings for each model are represented in Table 3. As meme is a combination of text and image, therefore we consider the majority voting based predictions as the final predictions for subtask 3.

System	Settings
MultiKernel LSTM with MLP	<ol style="list-style-type: none"> <li>1. <i>nb_filters</i>: 200</li> <li>2. <i>nb_rnnoutdim</i>: 600</li> <li>3. <i>rnn_dropout</i>: 0.5</li> <li>4. <i>optimizer</i>: adam</li> <li>5. <i>Threshold</i>: 0.3</li> <li>6. <i>Epochs</i>: 600</li> </ol>
Fine-tuned DistilBERT	<ol style="list-style-type: none"> <li>1. <i>Tokenizer</i>: distilbert-base-uncased</li> <li>2. <i>Dropout</i>: 0.25</li> <li>3. <i>Learning rate</i>: 1e-4</li> <li>4. <i>Batch size</i>: 16</li> <li>5. <i>maximum length</i>: 60</li> <li>6. <i>Epochs</i>: 30</li> </ol>
Fine-tuned ResNet50	<ol style="list-style-type: none"> <li>1. <i>Image Size</i>: (224,224,3)</li> <li>2. <i>Train Batch Size</i>: 32</li> <li>3. <i>Test Batch Size</i>: 16</li> <li>4. <i>Optimizer</i>: Adam</li> <li>5. <i>Optimizer Learning rate</i>: 2e-4</li> <li>6. <i>Epochs</i>: 900</li> </ol>

Table 3: System settings for Subtask 3.

### 3.3 Results and Analysis

We now compare our proposed CSECUDSG system’s performance with other participants systems in three subtasks as shown in Table 4, Table 5, and Table 6, respectively. In all the subtasks, the baseline system is set to random. The organizers used the F1-Micro as the primary evaluation measure for all the subtasks.

The overall scores of the three subtasks portray that our system acquired competitive performance. However, in all the subtasks, our system has some shortcomings with respect to the top performing teams. MinD, Volta, and Alpha are the top-performing teams in corresponding subtasks. We further analyze the performance of our systems in the subsequent section.

Team_Name	F1-Macro	F1-Micro
MinD	0.28993	0.59331
Alpha	0.26218	0.57187
Volta	0.26621	0.56958
<b>CSECUDSG</b>	<b>0.18454</b>	<b>0.48894</b>
NLPITR	0.12590	0.37917
TriHeadAttention	0.02397	0.18373
Baseline	0.04427	0.06439

Table 4: Comparative performance analysis on test set for Subtask 1.

Team_Name	F1-Score	Precision	Recall
Volta	0.48166	0.50061	0.46409
HOMADOS	0.40737	0.41206	0.40278
WVOQ	0.26787	0.24265	0.29894
<b>CSECUDSG</b>	<b>0.11983</b>	<b>0.07952</b>	<b>0.24303</b>
YNUHPCC	0.09111	0.18583	0.06035
Baseline	0.00952	0.03368	0.00554

Table 5: Comparative performance analysis on test set for Subtask 2.

Team_Name	F1-Macro	F1-Micro
Alpha	0.27315	0.58109
MinD	0.24389	0.56623
1213Li	0.22830	0.54860
<b>CSECUDSG</b>	<b>0.12117</b>	<b>0.51312</b>
LIIR	0.18807	0.49835
WVOQ	0.23957	0.47779
Baseline	0.05152	0.07062

Table 6: Comparative performance analysis on test set for Subtask 3.

### 3.4 Discussion

In this section, we discuss the contribution of each model’s performance against the combined system. For subtask 1, we showed the individual model’s performance on the test set in Table 7. From the table, we can see that the decision tree classifier achieved a score of 0.335 where the score is 0.426 and 0.480 in the case of the logistic regression classifier and DistilBERT model respectively. Analyzing this individual model’s score, we can say that we achieved the highest score from the DistilBERT model. After applying majority voting, our score increased to 0.008% and the final score is 0.48894 which means that the ensemble of three individual models can detect better than individual models.

In subtask 3, from the Table 8 we observe that the fine-tuned DistilBERT model provides a little better score than the majority voting based model. However, for the multi-modal task, both text and image contexts are important, therefore we consider the majority voting based model.

Tasks	Text/Image	Predicted labels	Gold labels
Subtask #1	AMERICAN EXPERIENCE\nTHE GREAT WAR\n	<ul style="list-style-type: none"> <li>❖ Flag-waving</li> <li>❖ Slogans</li> <li>❖ Loaded Language</li> </ul>	[ ]
Subtask #2	AMERICAN EXPERIENCE\nTHE GREAT WAR\n	<ul style="list-style-type: none"> <li>❖ AMERICAN EXPERIENCE\nTHE GREAT WAR\n: Loaded Language</li> <li>❖ AMERICAN EXPERIENCE\nTHE GREAT WAR\n: Loaded Language</li> <li>❖ CAN EX: Loaded Language</li> <li>❖ PERIENCE\nTHE GREAT WAR\n: Loaded Language</li> </ul>	[ ]
Subtask #3	AMERICAN EXPERIENCE\nTHE GREAT WAR\n + Meme:794_image_batch_2.png	<ul style="list-style-type: none"> <li>❖ Flag-waving</li> <li>❖ Slogans</li> <li>❖ Loaded Language</li> </ul>	<ul style="list-style-type: none"> <li>❖ Flag-waving</li> <li>❖ Glittering Generalities (Virtue)</li> <li>❖ Transfer</li> </ul>

Figure 5: Erroneous detection of persuasive techniques.

Method	F1-Score
CSECUDSG	0.48894
Performance of Individual Models	
–Logistic Regression	0.33585
–Decision Tree	0.42685
–Fine-tuned DistilBERT model	0.48064

Table 7: Individual model’s performance for Subtask 1.

Method	F1-Score
CSECUDSG	0.51312
Performance of Individual Models	
–MKCNN+LSTM and MLP model	0.36836
–Fine-tuned ResNet50 model	0.46449
–Fine-tuned DistilBERT model	0.52899

Table 8: Individual model’s performance for Subtask 3.

Further, we look into the reason behind the inaccuracy of multiple labels detected by our systems in all the subtasks. For this purpose, we have shown some examples in Figure 5. We noticed that due to the imbalance of labels in the dataset, our systems could not detect the labels which are present in less amount. As the percentage of these three labels i.e. ‘Loaded Language’, ‘Smears’, and ‘Name calling/Labeling’ are higher than the other labels, our system detects these three labels considerably but overlooks other labels.

#### 4 Conclusion and Future Directions

In this paper, we traversed different classification approaches along with a rich set of transfer learning features to tackle the challenges of the task. To predict the multiple labels in subtask 1 and 3, we exploited a unified architecture based on three different models. However, for span and technique classification in subtask 2, we used the pre-trained BERT model where the SemEval-2020 task 11 dataset is used to ameliorate the performance.

In the future, for subtask 1 and subtask 2, we have a plan to employ more pre-processing techniques and to conduct our experiment on efficient classifiers. Besides, we want to use deep learning models as well as other transfer learning fine-tuned models i.e. RoBERTa, BERT, GPT. For subtask 3, we plan to incorporate various image datasets to get more efficacious features.

#### References

- Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. 2016. Youtube-8m: A Large-Scale Video Classification Benchmark. *arXiv preprint arXiv:1609.08675*.
- Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, and Ion Androutsopoulos. 2019. Large-Scale Multi-label Text Classification on EU Legislation. *arXiv preprint arXiv:1906.02192*.
- Weiwei Cheng and Eyke Hüllermeier. 2009. Combining Instance-based Learning and Logistic Regres-

- sion for Multilabel Classification. *Machine Learning*, 76(2-3):211–225.
- Giovanni Da San Martino, Alberto Barrón-Cedeño, Henning Wachsmuth, Rostislav Petrov, and Preslav Nakov. 2020. *SemEval-2020 Task 11: Detection of Propaganda Techniques in News Articles*. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation (SemEval)*, pages 1377–1414, Barcelona (online). International Committee for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 4171–4186.
- Dimitar Dimitrov, Bishr Bin Ali, Shaden Shaar, Firoj Alam, Fabrizio Silvestri, Hamed Firooz, Preslav Nakov, and Giovanni Da San Martino. 2021. Task 6 at SemEval-2021: Detection of Persuasion Techniques in Texts and Images. In *Proceedings of the 15th International Workshop on Semantic Evaluation, SemEval '21*, Bangkok, Thailand.
- Hao Fei, Yue Zhang, Yafeng Ren, and Donghong Ji. 2020. Latent Emotion Memory for Multi-Label Emotion Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34,05, pages 7692–7699.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Dawid Jurkiewicz, Łukasz Borchmann, Izabela Kosmala, and Filip Graliński. 2020. ApplicaAI at SemEval-2020 Task 11: On RoBERTa-CRF, Span CLS and Whether Self-Training Helps Them. *arXiv preprint arXiv:2005.07934*.
- Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019. VisualBERT: A Simple and Performant Baseline for Vision and Language. *arXiv preprint arXiv:1908.03557*.
- Yikuan Li, Hanyin Wang, and Yuan Luo. 2020. A Comparison of Pre-trained Vision-and-Language Models for Multimodal Representation Learning across Medical Images and Reports. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1999–2004. IEEE.
- Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Haotang Deng, and Qi Ju. 2020. FastBERT: a Self-distilling BERT with Adaptive Inference Time. *arXiv preprint arXiv:2004.02178*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
- Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. *arXiv preprint cs/0205028*.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*.
- Gaku Morio, Terufumi Morishita, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 Task 11: An Empirical Study of Pre-Trained Transformer Family for Propaganda Detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1739–1748.
- F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research (JMLR)*, 12:2825–2830.
- S Rasoul Safavian and David Landgrebe. 1991. A Survey of Decision Tree Classifier Methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter. *arXiv preprint arXiv:1910.01108*.
- Lingyun Song, Jun Liu, Buyue Qian, Mingxuan Sun, Kuan Yang, Meng Sun, and Samar Abbas. 2018. A Deep Multi-Modal CNN for Multi-Instance Multi-Label Image Classification. *IEEE Transactions on Image Processing*, 27(12):6025–6038.
- Shiping Wen, Weiwei Liu, Yin Yang, Pan Zhou, Zhenyuan Guo, Zheng Yan, Yiran Chen, and Tingwen Huang. 2020. Multilabel Image Classification via Feature/Label Co-projection. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*.
- Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. <https://github.com/facebookresearch/detectron2>.
- Chenyang Zhang, Christine Kaeser-Chen, Grace Vesom, Jennie Choi, Maria Kessler, and Serge Belongie. 2019. The iMet Collection 2019 Challenge Dataset. *arXiv preprint arXiv:1906.00901*.

# UMUTeam at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with Linguistic Features and Word Embeddings

**José Antonio García-Díaz**

Facultad de Informática,  
Universidad de Murcia,  
Campus de Espinardo, 30100  
Murcia, Spain

joseantonio.garcia8@um.es

**Rafael Valencia-García**

Facultad de Informática,  
Universidad de Murcia,  
Campus de Espinardo, 30100  
Murcia, Spain

valencia@um.es

## Abstract

In writing, humor is mainly based on figurative language in which words and expressions change their conventional meaning to refer to something without saying it directly. This flip in the meaning of the words prevents Natural Language Processing from revealing the real intention of a communication and, therefore, reduces the effectiveness of tasks such as Sentiment Analysis or Emotion Detection. In this manuscript we describe the participation of the UMuTeam in HaHackathon 2021, whose objective is to detect and rate humorous and controversial content. Our proposal is based on the combination of linguistic features with contextual and non-contextual word embeddings. We participate in all the proposed subtasks achieving our best result in the controversial humor subtask.

## 1 Introduction

In this manuscript we describe our participation in the shared task *HaHackathon 2021* (Meaney et al., 2021), proposed in the Forum for Information Retrieval Evaluation (IberEval'2021) whose objective is the identification and evaluation of humorous and offensive texts written in English. Humor allows you to present the reality by highlighting the comic and ridiculous side of life. Humor is hard to identify, even for humans (Vrticka et al., 2013). On the one hand, there are many forms of humor: from mild forms, such as jokes or puns, that result in better and safer social environments, to biting forms, like sarcasm, which is used as a rhetorical device. Humor can also have a constructive end, as happens in satire, where irony, double meaning, and sharp analogies are used to ridicule someone or something. On the other hand, the sharper the humor, the more effort it takes to understand it. When humor is misunderstood, or when humor itself has a transgressive purpose, it can lead to controversies and confrontations. Furthermore, an added

difficulty is that humor is highly subjective and context dependent, making it even more difficult to understand (Jiang et al., 2019). Nevertheless, the benefits of endowing to a machine basic notions about humor and figurative language understanding outweigh the challenges they pose, because they lead to natural language-based interfaces to feel more naturally, such as chat-bots and virtual assistants (Ritschel et al., 2019) and more reliable results in tasks such as opinion mining.

We participate in all the proposed subtasks of HaHackathon'2021 with two systems that combines linguistic features (LF) extracted with a tool developed by our research group with (1) pre-trained word embeddings (PWE) from fastText (Mikolov et al., 2017) and GloVe (Pennington et al., 2014), and (2) contextual word embeddings from BERT (Devlin et al., 2018) (CWE). Our best result was achieved in task 1c (controversial humor), in which we trained a classification neural network that distinguishes between *non-humor*, *humor*, and *offensive* but outputs a binary prediction which indicates whether a text is controversial or not. The code is available at <https://github.com/Smolky/hahackathon-2021>.

## 2 Background

The HaHackathon 2021 challenge consists in two binary classification problems of humorous content and controversial tweets and two regression problems to identify how humorous and controversial the annotators considered the texts. We only used the dataset given by the organizers, two pretrained word embeddings models, and the contextual word embeddings from BERT.

The dataset provided consisted in 10k tweets written in English and posted in three subsets, namely training, development, and testing, with a ratio of 80-10-10. All tweets were annotated by

US English-speaking annotators of different genders and age groups with the following questions: (1) *Is the intention of this text to be funny?*, (2) *[If so] How humorous do you find the text? (on a scale of 1 to 5)*, (3) *Is this text generally offensive?*, and (4) *[If so] How generally offensive is the text? (on a scale of 1 to 5)*. Based on the data we had during the competition, we observed that the training dataset was imbalanced for subtask 1a, with a predominance of humorous tweets (61.65%) and almost balanced for subtask 2a. For regression subtasks, the average rating was 2.26 ( $\sigma$  of 0.5670) for subtask 1b and 0.58 ( $\sigma$  of 0.98) for subtask 2a.

Most of the literature found on the detection of humor highlights the importance of figurative language in which, contrary to the literal sense of language, words and expressions change their conventional meaning to refer to something without saying it directly (del Pilar Salas-Zárate et al., 2020). The reader can find in that work a compendium of works that analyze sarcasm, irony, and satire in English. Modern approaches rely on novel deep-learning transformers, such as the work described in Javdan et al. (2020), focused on sarcasm from a figurative language perspective, and in which the authors used BERT to build an aspect-based sentiment analysis system to determine whether the response is sarcastic. Other works, such as the one described in del Pilar Salas-Zárate et al. (2017), explores the differences and similarities in how satire is perceived in countries that share the language, but not the cultural background. To do this, they compare the use of linguistic characteristics with a corpus of satirical news written in Spanish from Spain and another written in Spanish from Mexico.

### 3 System overview

Our proposal is based on the usage of LF, PWE, and CWE to detect humor and controversial content. CWE have outperformed previously state of the art results regarding text classification tasks. However, we state that both PWE and CWE ignore relevant clues that are present in writings. For example, we observed in the dataset provided, the presence of capital letters (commonly used when shutting or to raise the voice), quoted sentences or dialogues that reproduce real or figurative conversations that are not captured with any of the above techniques.

To obtain the LF, we use a subset of the features extracted with UMUTextStats (García-Díaz et al., 2020, 2021). This tool is inspired in LIWC

(Tausczik and Pennebaker, 2010) but designed by our research group from scratch for the Spanish language. As the HaHackathon 2021 dataset only deals with English, we only extract statistical features discarding all features that we had that work with Spanish lexicons. Specifically, we select:

- **Expressive lengthening**, drawing out or emphasizing a verbalized word, giving it character.
- **Common typing mistakes**, such as starting sentences in lowercase, numbers, consecutive repetitions of the same word, and incorrect use of punctuation.
- **Corpus statistics**, such as the standard type/token ratio (TTR), the number of words, syllables, sentences, quoted sentences, interrogative and exclamatory sentences, and the average length of the words.
- **Punctuation symbols and emoticons**.
- **Common traits used in social media communication**, such as the presence of hyperlinks, hashtags or jargon.

The next step was to obtain the best deep-learning architecture for each subtask. We evaluate two main approaches. On the one hand, we combine the LF with PWE and different deep-learning architectures. On the other, we combine the LF with CWE from BERT (*bert-base-uncased*) using HuggingFace<sup>1</sup>. After performing these two models, we sent our results to the platform to evaluate them with the development dataset, achieving our best result with BERT for subtask 1a and RNNs for subtasks 1b, 1c, and 2a. After performing these two models, we sent our results to the platform to evaluate them with the development dataset, achieving our best result with BERT for subtask 1a and RNNs for subtasks 1b, 1c, and 2a.

### 4 Experimental setup

Our experimental setup is depicted in Figure 1 and, in a nutshell, we can described as follows.

First, we perform a preprocessing stage that consist of:

1. Removing social media language, such as hyperlinks or mentions.

<sup>1</sup><https://huggingface.co/> (v3.4.0)

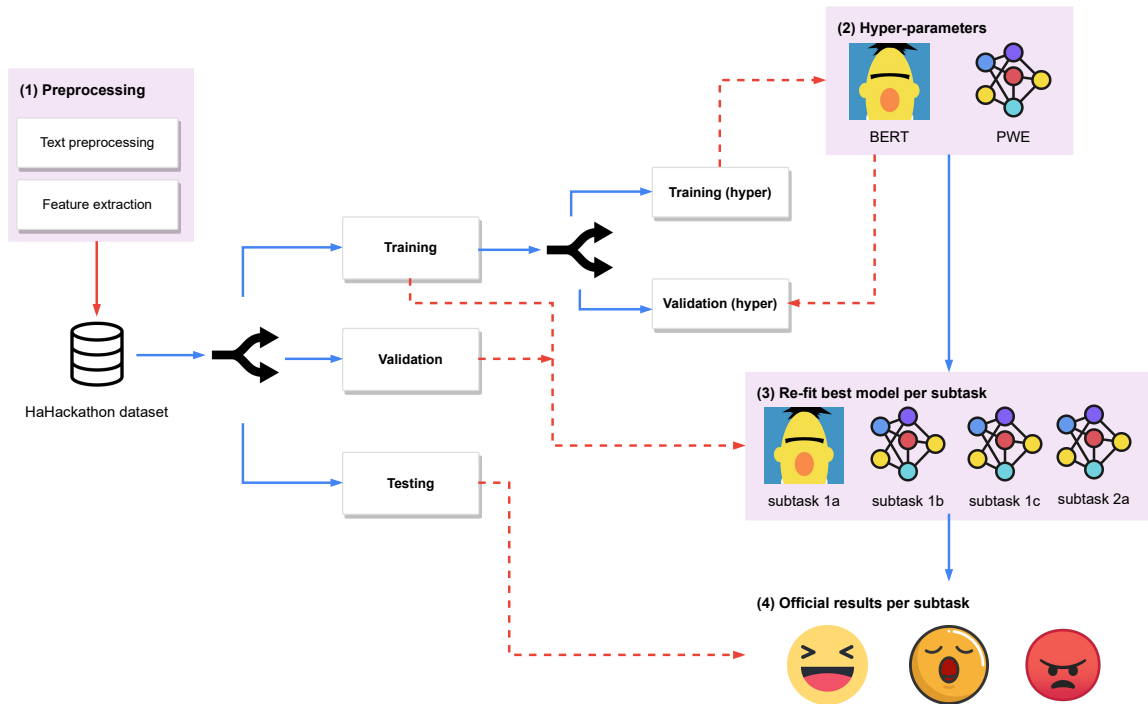


Figure 1: Pipeline of the participation of UMUTeam at HaHackathon'2021.

2. Removing digits.
3. Removing word elongations.
4. Transforming the tweets into their lowercase form.
5. Removing punctuation symbols.

Second, as the organizers of the shared task released the labels of the development dataset in the last stage of the competition, we begun the competition by dividing the 8K tweets of the training data into two folds in a ratio of 80-20. Third, two main approaches were evaluated. On the one hand, we use Talos<sup>2</sup> to evaluate different pretrained word embeddings models (FastText, GloVe, and word2vec) and several neural networks architectures (MLP, CNN, LSTM, BiLSTM, GRU, and BiGRU). On the other, we evaluate contextual word embeddings from BERT. These two processes are described in detail in the next paragraph. Finally, the classification subtasks (1a and 1c), the results are evaluated with the Accuracy and the F1-measure whereas the regression subtasks (1b and 2a) are evaluated with the Root Mean Squared Error (RMSE).

In case of BERT, we proceed as follows: we fine tune BERT with the training split of the Ha-

<sup>2</sup><https://github.com/autonomio/talos> (v0.6.6)

hackathon 2021 dataset for 2 epochs and a batch size of 64. Then, we freeze the resulting model and concatenate it to the LF in a new model composed of two hidden layers of 32 and 16 neurons respectively and trained for 10 epochs. In case of PWE, we evaluate word embeddings from fastText and GloVe but also we evaluate that the weights of the embeddings were learned from scratch in the embedding layer. Next, the LF and the word embeddings are concatenated, and used as input to a MLP, in which we evaluated (1) the number of hidden layers (between 1 and 8), (2) the number of neurons (8, 16, 48, 64, 128, 256), and (3) the shape of the network (*funnel*, *rhombus*, *long\_funnel*, *brick*, *diamond*, and *triangle*). We also evaluate the dropout rate to avoid overfitting (0, 0.2, 0.5, and 0.8), the activation function of the hidden layers (*relu*, *sigmoid*, *tanh*, *selu*, and *elu*), the learning rate, and the batch size (16, 32, and 64). Each combination of parameters was evaluated during 1000 epochs with an early stopping mechanism. Due to time constraints, we evaluated only 1000 combinations of these hyperparameters randomly selected.

## 5 Results

First, the best hyper-parameter combinations (validation set) are shown in Table 1.

Subtask 1a							
ACC	architecture	features	shape	# layers	1st neuron	dropout	PWE
83.926	MLP	lf+we	triangle	7	48	-	none
83.828	MLP	lf+we	long_funnel	7	16	-	fastText
83.809	CNN	lf+we	diamond	7	256	0.2	fastText
83.633	MLP	lf+we	brick	4	256	-	none
83.613	MLP	lf+we	rhombus	5	48	-	fastText
Subtask 1b							
RMSE	architecture	features	shape	# layers	1st neuron	dropout	PWE
0.79820	CNN	lf+we	funnel	2	256	-	gloVe
0.81080	BIGRU	lf+we	brick	5	64	0.5	fastText
0.81272	BILSTM	lf+we	brick	2	128	0.2	glove
0.81958	BILSTM	we	brick	3	48	0.2	glove
0.82004	LSTM	we	triangle	4	128	0.5	fastText
Subtask 1c							
ACC	architecture	features	shape	# layers	1st neuron	dropout	PWE
61.125	BILSTM	lf+we	triangle	1	48	0.5	gloVe
60.688	CNN	we	brick	2	48	-	gloVe
59.625	BILSTM	lf+we	triangle	4	48	0.2	gloVe
59.125	LSTM	we	triangle	3	256	0.5	fastText
59.000	LSTM	we	brick	5	8	0.5	none
Subtask 2a							
RMSE	architecture	features	shape	# layers	1st neuron	dropout	PWE
0.68037	CNN	we	triangle	4	128	0.5	fastText
0.68085	BIGRU	we	triangle	7	48	-	fastText
0.68399	LSTM	we	triangle	4	128	0.5	fastText
0.70100	CNN	lf+we	diamond	8	48	-	gloVe
0.70353	CNN	lf+we	funnel	2	256	-	gloVe

Table 1: Results of the best five hyperparameter combination for each subtask trained and evaluated with the training dataset with a ratio of 80-20.

We can observe that MLP and CNN perform better for subtask 1a, whereas recurrent neural networks achieve better results in subtasks 1b, 1c, and 2a. Regarding the feature sets, we observe that for subtasks 1a, 1b, and 1c, the combination of LF and WE achieved better results whereas only word embeddings achieved better results in subtask 2a. It draw our attention that the shape of the neural network (*shape*, *# layers*, and *1st neuron*) and the dropout rate vary according to the subtask. For example, in subtask 1a, four of the fifth best results were achieved without dropout, whereas for subtask 1b and 2a, a high dropout (0.5) resulted in better results.

Next, we compare our results with the rest of the participants and the baselines (see Table 2) with the test dataset. The organizers of the task provided two baselines based on a Naive Bayes for the classification subtasks (1a, 1c) and a Support Vector Regression for the regression subtasks (1b, 2a); both trained with a bag of words. In subtask 1a we achieve an F1-score of 91.6% and an accuracy of 93.25% reaching position 45. The best result is for *PALI* with an F1-score of 98.54% and an accuracy of 98.2%. In subtask 1b, we achieve an RMSE of 0.8847, reaching position 47 and falling below the baseline. The best result is for *abcbpc*, with an RMSE of 0.4959. For subtask 1c, we achieve an F1-score of 57.22% and an accuracy of 46.5%, reaching position 14. The best result is for *PALI*, with an F1-Score of 63.02% and an accuracy of 49.43%. In subtask 2a we achieved an RMSE of 0.8740, reaching position 46 and falling below the baseline. The best result was for *DeepBlueAI* with an RMSE of 0.412.

During the hyper parameter tuning stage, we evaluated the reliability of the PWE without the LF with our custom training and evaluating splits. The results in our development dataset were slightly better with the combination of both feature sets in three subtasks: 83.516% (LF+PWE) vs 83.379% (PWE) of accuracy in subtask 1a, 0.79820 vs 0.81958 of RMSE in subtask 1b, and 61.125% (LF+PWE) vs 60.688% (PWE) of accuracy in subtask 1c. However, in subtask 2a, PWE performed better without LF: 0.68037 (PWE) vs 0.70100 (LF+PWE).

## 6 Conclusions

While we are pleased with our participation since it has given us the opportunity to evaluate novel tech-

Subtask 1a			
#	Team	F1	Accuracy
1	PALI	98.54	98.20
2	stce	98.54	98.20
3	DeepBlueAI	96.76	96.00
4	SarcasmDet	96.75	96.00
<b>45</b>	<b>UMUTeam</b>	<b>91.60</b>	<b>93.25</b>
53	baseline	88.40	88.57
Subtask 1b			RMSE
#	Team		
1	abcbpc		0.4959
2	mmmm		0.4977
3	Humor@IITK		0.5210
4	YoungSheldon		0.5257
46	baseline		0.8609
<b>47</b>	<b>UMUTeam</b>		<b>0.8847</b>
Subtask 1c			
#	Team	F1	Accuracy
1	PALI	63.02	49.43
2	mmmm	62.79	46.99
3	SarcasmDet	62.70	46.99
4	ThisIstheEnd	62.61	46.02
<b>14</b>	<b>UMUTeam</b>	<b>57.22</b>	<b>46.50</b>
31	baseline	46.24	43.74
Subtask 2a			RMSE
#	Team		
1	DeepBlueAI		0.4120
2	mmmm		0.4190
3	HumorHunter		0.4230
4	abcbpc		0.4275
42	baseline		0.6415
<b>46</b>	<b>UMUTeam</b>		<b>0.8740</b>

Table 2: Comparison of our results with other participants and the baseline for each subtask

niques and to improve our methods, we consider our results to be far from competitive. It should be noted that, for reasons unrelated to this competition, we did not have time to do all the tests we wanted. On the one hand, the final labels of the development dataset were published before the final stage, but we did not fit the models with this new information. On the other hand, we only submit one run in the final stage. Compared with the rest of the participants, we observe they submitted an average of 7.5737 runs ( $\sigma$  of 8.1720). However, although we believe that our results could have been somewhat better, there is still a long way to go. First, it caught our attention that BERT does not outperform the results achieved with dense,



recurrent, and convolutional neural networks for subtasks 1b, 1c, and 2a. At this respect, we will review our pipeline to detect weakness. It also draw our attention that our results did not beat the baselines in the regression tasks which indicates some kind of implementation or conceptualization error. Second, we only evaluated a subset of the LF that we had for Spanish. Accordingly, we will adapt UMUTextStats to English and compare their reliability with de-facto tools like LIWC. Third, we will focus on the interpretability of the models, as we believe that LF result in more interpretable models. Finally, we will evaluate machine learning ensembles as a mean of combining the LF.

## Acknowledgments

This work was supported by the Spanish National Research Agency (AEI) through project LaTe4PSP (PID2019-107652RB-I00/AEI/10.13039/501100011033). In addition, José Antonio García-Díaz has been supported by Banco Santander and University of Murcia through the industrial doctorate programme.

## References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- José Antonio García-Díaz, Mar Cánovas-García, Ricardo Colomo-Palacios, and Rafael Valencia-García. 2021. Detecting misogyny in spanish tweets. an approach based on linguistics features and word embeddings. *Future Generation Computer Systems*, 114:506–518.
- José Antonio García-Díaz, Mar Cánovas-García, and Rafael Valencia-García. 2020. Ontology-driven aspect-based sentiment analysis classification: An infodemiological case study regarding infectious diseases in latin america. *Future Generation Computer Systems*, 112:641–657.
- Soroush Javdan, Behrouz Minaei-Bidgoli, et al. 2020. Applying transformers and aspect-based sentiment analysis approaches on sarcasm detection. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 67–71.
- Tonglin Jiang, Hao Li, and Yubo Hou. 2019. Cultural differences in humor perception, usage, and implications. *Frontiers in psychology*, 10:123.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Tomás Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhresch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *CoRR*, abs/1712.09405.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- María del Pilar Salas-Zárate, Giner Alor-Hernández, José Luis Sánchez-Cervantes, Mario Andrés Paredes-Valverde, Jorge Luis García-Alcaraz, and Rafael Valencia-García. 2020. Review of english literature on figurative language applied to social networks. *Knowl. Inf. Syst.*, 62(6):2105–2137.
- María del Pilar Salas-Zárate, Mario Andrés Paredes-Valverde, Miguel Ángel Rodríguez-García, Rafael Valencia-García, and Giner Alor-Hernández. 2017. Automatic detection of satire in twitter: A psycholinguistic-based approach. *Knowl. Based Syst.*, 128:20–33.
- Hannes Ritschel, Ilhan Aslan, David Sedlbauer, and Elisabeth André. 2019. Irony man: Augmenting a social robot with the ability to use irony in multimodal communication with humans. In *AAMAS '19*, page 86–94, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.
- Pascal Vrticka, Jessica M Black, and Allan L Reiss. 2013. The neural basis of humour processing. *Nature Reviews Neuroscience*, 14(12):860–868.

# ES-JUST at SemEval-2021 Task 7: Detecting and Rating Humor and Offensive Text Using Deep Learning

**Emran Al Bashabsheh**

Computer Science Department  
Jordan University of  
Science and Technology  
Irbid, Jordan

emranalbashabsheh@gmail.com

**Sanaa Abu Alasal**

Computer Science Department  
Jordan University of  
Science and Technology  
Irbid, Jordan

sanaasal11@gmail.com

## Abstract

This research presents the work of the team's ES-JUST at semEval-2021 task 7 for detecting and rating humor and offensive text using deep learning. The team evaluates several approaches (*i.e.* BERT (Devlin et al., 2018), Roberta (Liu et al., 2019), XLM-Roberta (Conneau et al., 2019), and BERT embedding + Bi-LSTM) that employ in four sub-tasks. The first sub-task deal with whether the text is humorous or not. The second sub-task is the degree of humor in the text if the first sub-task is humorous. The third sub-task represents the text is controversial or not if it is humorous. While in the last task is the degree of an offensive in the text. However, Roberta pre-trained model outperforms other approaches and score the highest in all sub-tasks. We rank on the leader board at the evaluation phase are 26, 26, 25, and 9 through 0.9564 F-score, 0.5709 RMSE, 0.4888 F-score, and 0.4467 RMSE results, respectively, for each of the first, second, third, and fourth sub-task, respectively.

## 1 Introduction

Dealing with natural languages has long been a challenge and an interesting topic for researchers (Chowdhury, 2003). Understanding and generating languages is part of natural language processing (NLP) (Nadkarni et al., 2011). Recently, the language model is able to deal with sequence-to-sequence problems such as question and answer, translation, multiple choice. In addition, it is able to capture complex relationships, semantic meaning, word meaning disambiguation, and word aspect-based (Deng and Liu, 2018). Humorous text is one of the important things we watched every day. It is commonly used to express an opinion on issues (societal, political, sports, and economic), whether in posts on social media platforms, or as advertising for a specific product (Kramer, 2011). In addition, the humor in the text makes the text complex in

terms of interpretation and understanding of the text. Because of the manipulation of the meaning of words and the way the text is written to express the sense of humor in the words. On the other hand, understanding the humorous in the text varies according to the age or gender of the person, or even according to the culture, social status and mentality of the person (Goel and Dolan, 2007). In this task, a dataset was collected in the English language that represents humor and joke in the text and words. We participated in this task to build an approach capable of distinguishing a text that is humorous or not. Here we have explicitly used pre-trained models that deal with the concept of contextual text such as Bert (Devlin et al., 2018), Roberta (Liu et al., 2019), and XLM-Roberta (Conneau et al., 2019). In addition, as a baseline we worked on training the dataset by the Bert embedding layer and extracting weights to feed it into the Bi-LSTM and Dense layers.

In all sub-tasks we used as a baseline BiLSTM (Graves and Schmidhuber, 2005) layer with a BERT embedding layer, as well as, pre-trained models such as Bert (Devlin et al., 2018), Roberta (Liu et al., 2019), and XLM-Roberta (Conneau et al., 2019).

In all sub-tasks, Roberta model showed superiority compared to other approaches. We ordered according to the official results among 36 participating teams. In the first sub-task, we achieved 26th rank with an 0.9564 F-score result. On the second sub-task, ranked 26th with a score of 0.5709 RMSE. A third sub-task placed 25th with 0.4888 F-score (Sokolova et al., 2006). The last sub-mission we took the 9th rank with a score of 0.4467 RMSR (Chai and Draxler, 2014). The remainder of this paper is organized as follows: Background in Section 2. The properties of the dataset and the system in section 3. Section 4 explains the experiment and analyzing results. The last section 5 shows

conclusions and future work.

## 2 Background

In (Hossain et al., 2019) developed a new humor corpus, which consist of 15,095 news headlines in English. They substituted the headlines with few words to be funny. Also, (Li et al., 2020) used attention-based with bi-directional long short-term memory (AttBiLSTM) to classify slang language into negative humor or positive humor. In (Anamoradnejad, 2020) utilized a BERT embedding layer with several parallel hidden layer to categorize 200K humorous sentences whether (positive or negative). While (Fan et al., 2020) used two kinds of attention mechanisms (internal and external) to capture sense of humor in words. Most of the previous works came to predict the humor polarity (positive, negative) or the humor rating (range values) in the text. However, this research addressing the humor and offensive score detection.

## 3 Methodology

### 3.1 Task Description

We worked with four sub-tasks provided by SemEval-2021 <sup>1</sup>, in task-1 divided into (a, b and c sub-tasks). Each sub-task related to the other. Moreover in task-2 has one sub-task (a). In general, Sub-task-1a will predict whether the text expresses a humorous or not (binary classification problem 1, 0). Sub-task-1b if the text is considered a humorous, will predict how humorous it is from 0 to 5 values (regression problem). Sub-task-1c If the text is a humorous, we would predict if it is controversial or not (binary classification problem 1, 0). Sub-task-2a will predict the offense.

### 3.2 Data-set

A dataset consists of a set of texts and each text has four categories (is-humor, humor-rating, humor-controversy, offense-rating) in English language (Meaney et al., 2021). Each text asked by 20 annotators to label each category of the text. As well as, the annotators come from different gender and age groups. For is-humor and humor-controversy categories were taken the majority of the classes by 20 annotators as label for each text. Whereas, humor-rating and offense-rating categories take the average of rating classes between range 1 and 5 over 20 annotators as a label for each text. Table 2

<sup>1</sup>[https://competitions.codalab.org/competitions/27446#learn\\_the\\_details](https://competitions.codalab.org/competitions/27446#learn_the_details)

shows examples of the training dataset per text with the four classifications for each category. Moreover, in humor-rating and humor-controversy, we noted the categories have many NaN values, because if is-humor the majority of the classes were not classified as humor which means 0 label, so the remaining categories of humor are NaN values. Therefore, we need to remove the NaN values from each category as pre-processing the dataset before training the models. Table 1 shows the total number in the training, development and testing dataset for each category.

Dataset	Is-H	H-R	H-C	O-R
Training	8000	4932	4932	8000
Development	1000	632	632	1000
Testing	1000	615	615	1000

Table 1: The total number for each category (is-humor, humor-rating, humor-controversy, offense-rating) after removing NaN values.

### 3.3 System overview

The proposed system focused on pre-trained transformer models, we Moreover applied some techniques that represent embedding words and feeding them into long short-term memory (LSTM) layers to train the data-set. Through all of the sub-tasks, the highest score was via the Roberta model. It is one of the powerful models pre-trained on a huge data-set and complex architecture. As well, it was released by Facebook and designed base on the BERT model that was released by Google. All pre-trained models are capable of handling long text dependencies and capturing features and relationships. Furthermore, the structure of pre-trained models that involve encoder-decoder (Cho et al., 2014) is enabled to deal with sequence-to-sequence (Sutskever et al., 2014) tasks. In addition, BERT-Large (Devlin et al., 2018) and Roberta-Large (Liu et al., 2019) models consisted of 24 layers, 1024 hidden units of output word embedding, and 16 head attention layers, where both models have the same layered structure but differ in the method of approach to training and the volume of data used to train each model.

There are two approaches used to train a BERT model 1- Masked Language Model (MLM) is masking some tokens of the training dataset with a [mask] symbol and try to predict the token. 2 - Next Sentence Prediction (NSP) is training the dataset by assigning 1's for neighboring sentences

Text	Is-H	H-R	H-C	O-R
TENNESSEE: We're the best state. Nobody even comes close.	1	2.42	1	0.2
*Elevennessee walks into the room* TENNESSEE: Oh shit				
I got REALLY angry today and it wasn't about nothing, but you're going to have to take my word for that.	0	Nan	Nan	0.15
Told my mom I hit 1200 Twitter followers. She pointed out how my brother owns a house and I'm wanted by several collection agencies. Oh ma!	1	2.11	1	0

Table 2: An example illustrating the features of a training dataset whether a humor or offense. If Is-humor class is 0 then the Humor-rating and Humor-controversy classes are Nan values.

and 0's for randomly chosen sentences. In contrast, Roberta used the MLM model approach for the training phase, as well as trained on a huge dataset compared to the BERT model.

Moreover, we tried the XLM-Roberta-Large pre-trained model (Conneau et al., 2019), which has 550M parameters with 24-layers of architecture. In addition, it consists of 1024 of the output hidden-state embedding, 4096 of feed-forward hidden-state, and 16 of head attentions. The model has Trained on 2.5 TB of newly created clean CommonCrawl data that supports 100 languages.

On the other hand, this research exploits BERT embedding to represent text. Where the weights were extracted by training the dataset on the BERT embedding layer and then feeding them into a BiLSTM layer of 128 units (Graves and Schmidhuber, 2005). Moreover, We used the dropout layer with 0.3 ratios, the max-pool layer, then passing the information into a dense layer with 64 units. In the last layer for classification tasks, the final dense layer is 2 hidden output units with a sigmoid activation function, and for regression, one unit output in the final dense layer. The Figure 1 shows the model architecture used for prediction label on classification and regression tasks.

## 4 Experimental and Results

In the experimental phase, the dataset was divided into three parts (training, development, and testing). We used the training dataset to train the model, and the development dataset to fine-tune the model to capture the best hyper-parameters without occurring over-fitting or under-fitting the model. Moreover, we used the test data set to check the performance of the model with an unseen dataset and to ensure the generalizability of the model. However, to perform the experiments we used collaborative google Colab as a platform, which provides a num-

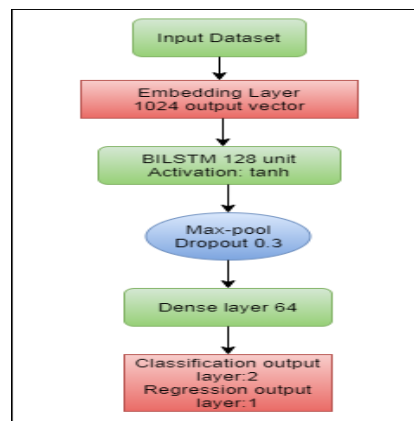


Figure 1: An illustration of the proposed model architecture.

ber of GPUs available for use with modest memory size<sup>2</sup>. In addition, in our experiments with pre-trained models, we used the transformers library that is based on the PyTorch language and allows you to fine-tune the models and train them on your own dataset<sup>3</sup>. In training the model, we did not use any pre-processing technique in the entered dataset. Although, there are some symbols, upper and lower case letters, misspellings, and some abbreviations in the dataset. However, We did not treat these issues in dataset, where the dataset is trained as it is. In order, for the model to be more realistic and robust in dealing with the real dataset. As well as, the model might deal with those cases as features for each case in the dataset for the model learning phase. Just in pre-processing phase, we needed to remove NaN values in both sub-task (1B and 1C). In order to test the performance of the approaches used in this task, where each sub-task has a metric that meets the type of output of each sub-task such as regression metric or classification metric. Accuracy and F-score metrics were a measure of the

<sup>2</sup><https://colab.research.google.com>

<sup>3</sup><https://huggingface.co/transformers/>

performance in sub-task-1a and sub-task-1c. Likewise, the RMSE metric was a measure of outcome performance in both sub-task-1b and sub-task-2c. In the process of model tuning, we tried several hyper-parameters, where the batch size was fixed 8, and the Adam optimizer function was used on all experiments. Furthermore, we applied several learning-rates in the range 1e-5, 4e-5, 1e-6, 3e-6 and a different number of epoch 2, 4, 8, 12 epochs. The table 3 shows the main experiments among many models with different LRs and Epochs for each sub-task.

Sub-task	Model	Epoch	LR
1-A	Roberta-Large	4	1e-6
	Roberta-Large	4	3e-6
	Roberta-Large	8	1e-6
	XLM-Roberta-Large	8	1e-6
	BERT-Large-Cased	8	1e-6
	BERT embedding + BiLSTM	ES	2e-5
1-B	Roberta-Large	8	1e-6
	Roberta-Large	8	1e-5
	Roberta-Large	12	1e-5
	Roberta-Large	2	3e-5
	BERT-Large-Cased	8	3e-5
	BERT embedding + BiLSTM	ES	2e-5
1-C	Roberta-Large	8	1e-6
	Roberta-Large	8	1e-5
	Roberta-Large	8	8e-5
	XLM-Roberta-Large	8	8e-5
	BERT-Large-Cased	8	8e-5
	BERT embedding + BiLSTM	ES	2e-5
2-A	Roberta-Large	8	1e-5
	Roberta-Large	8	3e-5
	Roberta-Large	4	1e-5
	Roberta-Large	12	1e-5
	BERT-Large-Cased	8	1e-5
	BERT embedding + BiLSTM	ES	2e-5

Table 3: The models applied and hyper-parameters used. (ES denotes to early-stopping technique)

## 4.1 Result

Roberta achieved high-performance results compared to other approaches, that exhibit his ability to capture traits and distinguish between labels. The table 4 presents the best results for both development and evaluation level results, as well as the best hyper-parameters selected based on the experimental phase for each sub-task. In sub-task-1A Roberta-

Large achieved high scores in a binary classification problem compared to the other models, where we scored 26 at F-score metrics in the evaluation phase for our ranking on the leader-board. While in the sub-task-2B also Roberta achieved acceptable results in the regression problem and outperformed the other models, as we ranked on the Leader Board 26 at RMSR metric. For the rest of the other sub-tasks, sub-task -3C is treated as a binary classification, which we achieved 25 rank in evaluation phase by F-score. In the last sub-task, our rank was 9 for an RMSR metric at the evaluation phase on the leader board.

### 4.1.1 Error Analysis

This section presents some analyzes to clarify the outcomes and limitations model of each sub-task. Figure 2 represents the confusion matrix for each label is given in sub-task-1A which is a classification problem. The figure shows the number of cases which the actual label matches the predicted label ( $y = \hat{y}$ ) which is 946 in total. While the number of labels that differ ( $y \neq \hat{y}$ ) that the model could not predict the label, it is 54. In the square that represents 31 false positives, we can see that it is a little more than the square that represents 23 false positives. This is because the training dataset is slightly biased towards label 1, which is 4,932 out of 8,000, while label 0 makes up 3,068 of the training data set.

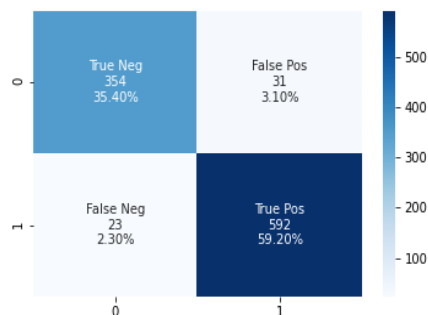


Figure 2: An illustration of the confusion matrix for sub-task-1A.

Moreover, the sub-task-1B represented in figure 3. We applied the round function to obtain integer numbers and categories of labels to display, which shows four values existing in this task as continuous labels in range 0 - 4. In the figure, the label shows a label 2 obtained the highest match in the model between the actual labels and the predicted

Sub-task	Model	Epoch	LR	Development result	Evaluation result
1-A	Roberta-L	8	1e-6	0.9426-F1 & 0.9270-Acc.	0.9564-F1 & 0.9460-Acc.
1-B	Roberta-L	2	3e-5	0.518-RMSE	0.5709-RMSE
1-C	Roberta-L	8	8e-5	0.5493-F1 & 0.5585-Acc.	0.4888-F1 & 0.5545-Acc.
2-A	Roberta-L	12	1e-5	0.5209-RMSE	0.4467-RMSE

Table 4: The best results gained for both development and evaluation level with hyper-parameters chosen.

labels, while the labels 0, 1, 4, the model could not recognize them in the prediction phase. This is due to the size of the training dataset is a little for each label compared to 2, 3 labels. The size of the dataset in 0, 1, and 4 labels in the training dataset are 16, 410, and 47 respectively. On the other hand, label 2 is repeated 2835 times and 3 is repeated 1624 times in the training dataset.

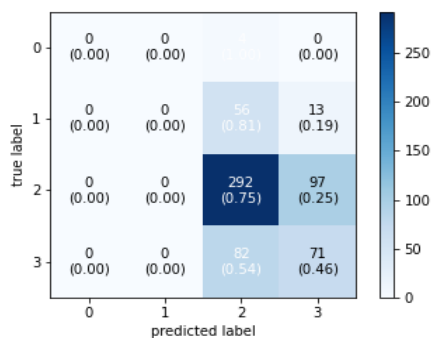


Figure 3: An illustration of the discretization confusion matrix for sub-task-1B.

A third sub-task, which is a binary classification problem. The figure 4 shows the model is able to recognize label 0 a little more than label 1, but in general, the model is not learned well (high biased). The number of cases for both (0 and 1) labels in the training dataset were 2467 and 2465 almost equal, respectively.

Finally, in the last sub-task-2C, we needed to use a round function to approximate continuous values to discrete values. However, the diameter of the figure 5 clearly shows the highest label to the lowest label distinguished by the model. Where the values are logically acceptable compared to the number of cases for each label in the training dataset, which are 5737, 1043, 623, 364, 214, and 19 frequency for each of 0, 1, 2, 3, 4, 5 labels respectively.

## 5 Conclusion

In this paper, we presented several approaches that addressed four sub-tasks. We obtained high scores

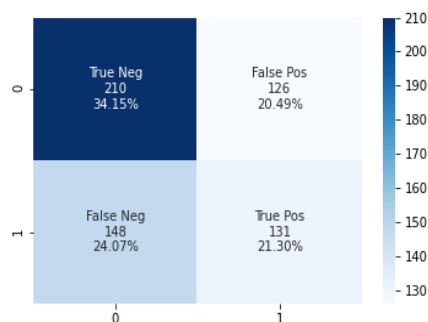


Figure 4: An illustration of the confusion matrix for sub-task-1C.

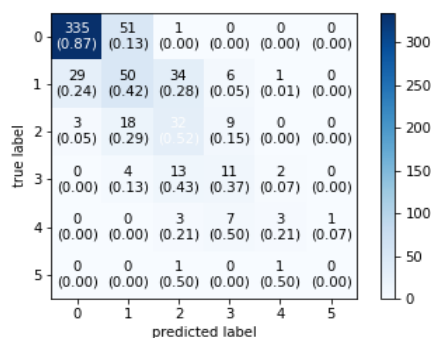


Figure 5: An illustration of the discretization confusion matrix for sub-task-2A.

using a pre-trained Roberta model for each sub-task. In the first sub-task, predicting if the text is humorous or not, we gained a 0.9564 F-score. While in the second sub-task, finding a humorous text representation rate from 0 to 5, that was got a 0.5709 RMSE. A third sub-task, verification of the text is controversial or not, obtained a 0.4888 F-score. The last sub-task is to find the offensive rate in the text for the range of 0 to 5, which achieved 0.4467 RMSE. For future works, we are going to do more experiments and using ensemble technique to enhance the results. Moreover, adding more dataset with the original to treat the biased label.

## References

- Issa Annamoradnejad. 2020. Colbert: Using bert sentence embedding for humor detection. *arXiv preprint arXiv:2004.12765*.
- Tianfeng Chai and Roland R Draxler. 2014. Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250.
- Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Gobinda G Chowdhury. 2003. Natural language processing. *Annual review of information science and technology*, 37(1):51–89.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.
- Li Deng and Yang Liu. 2018. *Deep learning in natural language processing*. Springer.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Xiaochao Fan, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Yanbo Zou. 2020. Humor detection via an internal and external neural network. *Neurocomputing*, 394:105–111.
- Vinod Goel and Raymond J Dolan. 2007. Social regulation of affective experience of humor. *Journal of cognitive neuroscience*, 19(9):1574–1580.
- Alex Graves and Jürgen Schmidhuber. 2005. Frame-wise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.
- Nabil Hossain, John Krumm, and Michael Gamon. 2019. ” president vows to cut taxes, hair”: Dataset and analysis of creative text editing for humorous headlines. *arXiv preprint arXiv:1906.00274*.
- Elise Kramer. 2011. The playful is political: The metapragmatics of internet rape-joke arguments. *Language in Society*, pages 137–168.
- Da Li, Rafal Rzepka, Michal Ptaszynski, and Kenji Araki. 2020. Hemos: A novel deep learning-based fine-grained humor detecting method for sentiment analysis of social media. *Information Processing & Management*, 57(6):102290.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. 2011. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551.
- Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. 2006. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer.
- Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

# Tsia at SemEval-2021 Task 7: Detecting and Rating Humor and Offense

Zhengyi Guan, Xiaobing Zhou\*

School of Information Science and Engineering Yunnan University, Yunnan, P.R. China

\*Corresponding author: zhouxb@ynu.edu.com

## Abstract

This paper describes our contribution to SemEval-2021 Task 7: Detecting and Rating Humor and Offense. This task contains two sub-tasks, sub-task 1 and sub-task 2. Among them, sub-task 1 contains three sub-tasks, sub-task 1a, sub-task 1b and sub-task 1c. Sub-task 1a is to predict if the text would be considered humorous. Sub-task 1c is described as follows: if the text is classed as humorous, predict if the humor rating would be considered controversial, i.e. the variance of the rating between annotators is higher than the median. We combined three pre-trained models with CNN to complete these two classification sub-tasks. Sub-task 1b is to judge the degree of humor. Sub-task 2 aims to predict how offensive a text would be with values between 0 and 5. We use the idea of regression to deal with these two sub-tasks. We analyze the performance of our method and demonstrate the contribution of each component of our architecture. We have achieved good results under the combination of multiple pre-training models and optimization methods.

## 1 Introduction

Humor is an intellectual activity that can cause certain emotions in human thinking. Humor is not only very important but also very common in daily life. People's research on humor has involved many fields such as psychology, sociology, linguistics and so on. Of course, it also has special value for the research of computing languages. Because of its complexity and inherent subjectivity, the development of automatic humor recognition and assessment poses a great challenge in Computational Linguistics, and therefore is a popular subject in various shared task competitions. (Dick et al., 2020) However, we must also recognize the difficulties of humor research. First of all, although humans can easily judge whether a sentence is humorous

in daily life, but due to humor is restricted by geography, environment, social background and other aspects, we usually not only pay attention to this sentence or whether the matter is humorous, We have to figure out how humorous or funny this context is? In other words, we pay more attention to its degree of humor which is not so easy for computers. And because humor is affected by the environment, different people have different understanding of humor. Just like sometimes your humor is based on the suffering of others. Things you find funny, but others don't necessarily find them funny. In other words, humor is controversial. So we have to determine the specific humor rate. This task is to take a median as the criterion for humor. It is also difficult to judge the humor of this dispute by computer language.

More recently, some humorous sentences can also have derogatory and offensive elements. Whether humor can cause offense is also one of the researches in this thesis. I believe that the study of humor not only helps to improve the computer's understanding of humor in certain aspects, but also purifies our network environment.

The four sub-tasks of SemEval-2021 task7 are designed to solve the above problems. For deep learning, the computer must not only judge whether a sentence is humorous. It is more important to understand this humorous sentence. In our paper, we designed two effective systems to solve the above four sub-tasks. For Sub-task 1a and Sub-task 1c, We take them as a binary classification task. We designed an efficient system using the idea of BERT-CNN. This is not a new idea because people have tried in the past. We also use the other popular pre-training models, including the derived ALBERT and RoBERTa. For sub-task 1a, we need to judge whether a sentence is humorous. We predict a Label L: where  $L \in \{is\_humor - 1, not\_is\_humor - 0\}$ . For sub-task 1c, we also need to judge whether a sentence is controversial and predict a Label L: where  $L \in \{humor\_controversy - 1, not\_humor\_controversy - 0\}$ . For sub-task 1b,



we combined regression ideas with the current popular pre-training model to complete these two sub-tasks. The two input sentences were split into two lists and fed into the Regression Model, which made a prediction about the funniness of each sentence. Then we compared the results of the prediction to determine the funnier of the two sentences. We compare the humor between each sentence and finally return a humor value. (Ammer and Grüner, 2020) Finally, the humor rate and the level of controversy are mapped to the range from 0 to 5. We use mean square error to measure these two tasks.

## 2 Background

The judgment of humor is the same as other text classification problems in natural language processing. The most important thing is to find suitable features to represent sentences. The task is to assign predefined categories to a given text sequence. Many works have shown that pre-trained models on large corpora are beneficial for text classification and other NLP tasks, which can avoid training new models from scratch. Since 2013, people have proposed some word embedding approaches such as word2vec (Mikolov et al., 2013) and glove (Pennington et al., 2014). However, because their word embeddings are all in the same space, they cannot express the role of polysemy. In other words, they are non-contextual embeddings, they cannot capture the high-level concepts of sentences, such as semantics and context (Sun et al., 2020). Later, someone proposed the ELMo model to solve this problem. Compared with word2vec and glove, ELMo captures contextual information and not just individual information of words. In word2vec, the vector representations of words are completely consistent in different contexts, and ELMo is optimized for this (Zhang et al., 2017). More recently, pre-trained language models have shown to be useful in learning common language representations by utilizing a large amount of unlabeled data: such as OpenAI GPT (Brown et al., 2020) and BERT (Devlin et al., 2018). BERT is based on a multi-layer bidirectional Transformer (Vaswani et al., 2017) and is trained on plain text for masked word prediction and next sentence prediction tasks. This paper tried other two new pre-training models of ALBERT (Lan et al., 2019) and RoBERTa (Liu et al., 2019) based on BERT. And we fine-tuned down-

stream tasks for a variety of pre-trained models. Finally, we completed these four sub-tasks effectively.

## 3 System overview

### 3.1 Data

The data of the four sub-tasks are all provided by SemEval (Meaney et al., 2021). The official organizer provides the same training set and test set for all sub-tasks. We split the training set into a new training set and a test set by using the stratified 5-fold cross-validation. BERT uses the wordpiece tool for word segmentation and inserts special separators (`[CLS]` which are used to separate each sample) and separator (`[SEP]` which are used to separate different sentences in the sample). For each fold of the data set, the input data format is as follows: `[CLS]+[sentence]+[SEP]` (Bai and Zhou, 2020). There are a total of 8000 data in the training set and 1000 data in the test set. In addition, the training set was split into 85% and 15% for training and development set respectively (Note: We did not include the use of the development dataset which was given by the task organizers). They are all English sentences. Each sentence of the data in the training set can be regarded as a combination of `{id,text}` and one of `{is_humor,humor_rating,humor_controversy,offense_rating}`. For the data we did a simple pre-processing. We first remove the characters specified at the beginning and end of the string. (the default is a space or a newline.)

### 3.2 Methodology

Text classification technology is an efficient information retrieval and data mining information technology. The classification method based on machine learning trains a classification model by learning a given training set, and then uses the training model to classify text. Traditional machine learning classification methods include: random forest (RF), naive Bayes (NB), logic Regression (LR) and Support Vector Machine (SVM), etc. However, with the development of deep learning, many NLP tasks can adopt a pre-training + fine-tuning structure. The most typical is the BERT pre-training model. We propose two architectures to solve four sub-tasks.

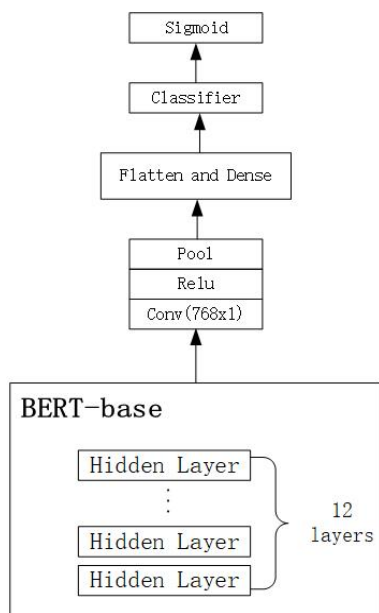


Figure 1: Model for Sub-task 1a and Sub-task 1c

### 3.2.1 Method A

Method A is designed to solve sub-task 1a and sub-task 1c. CNN for textual tasks by Kim (Kim, 2014) showed superiority in text classification tasks. CNN can be used with learned vector representations of the text (embeddings). These embeddings may either be initialized randomly and trained along with the model, or can be pre-trained vectors.

The proposed model maximizes the utilization of knowledge embedded in pre-trained BERT language models by feeding the outputted contextualized embeddings of its last four hidden layers into a several filters and convolution layers of the CNN. Finally, the output of the CNN was passed to a dense layer and the predictions were obtained (Safaya et al., 2020).

As shown in Figure 1, we use BERT-base as a pre-training model to build the model and other pre-training models are similar. BERT is a model built based on Transformer Encoder. Its entire architecture is actually based on DAE (Denoising Autoencoder). This part is called Masked Language Model (MLM) in the BERT (Devlin et al., 2018) article. MLM is not strictly a language model, because the entire training process is not trained using a language model. BERT randomly replaces some words with the MASK tag, and then predicts the word masked. The process is actually the process of DAE. BERT has two main trained models, namely BERT-Small and BERT-large. BERT-large uses a 12-layer encoder structure, that is, twelve

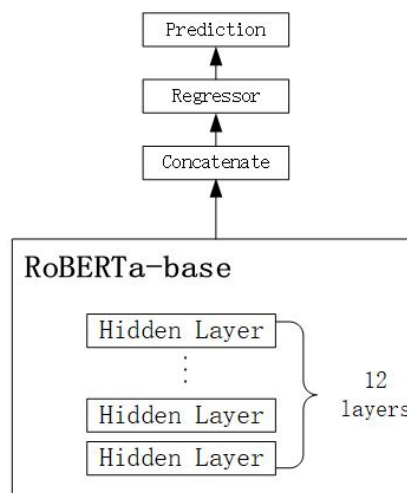


Figure 2: Model for Sub-task 1b and Sub-task 2

hidden layers. The whole model has a lot of parameters. For sub-task 1a and sub-task 1c, we tried a variety of methods based on BERT-base, including BERT+LSTM and other pre-trained models ALBERT and RoBERTa (add a linear layer). The last method to try is after BERT pre-training model, we use one or two layers of CNN to perform feature extraction. Finally, we input into a linear classifier to classify English sentences (humorous or controversial).

### 3.2.2 Method B

Method B is used to solve sub-task 1b and sub-task 2 (Regression tasks). Since the values we want to output (values between 0 and 5) are continuous. We pre-trained through the input of two English sentences and then made a humorous (controversial) comparison. Since the effect of CNN on the regression task is not very useful, we mainly tried and improved on the pre-training model. We mainly use BERT, ALBERT and RoBERTa for word embedding. As shown in Figure 2, RoBERTa works best. BERT has the worst effect. Because RoBERTa is trained with dynamic masking, FULL SENTENCES without NSP loss, large mini-batches and a larger byte-level BPE (Liu et al., 2019). In addition, it adjusted the parameters of the Adam algorithm. From 16G data to 160G. RoBERTa uses a larger batch size, and the number of training is more. The network structure is complex, so the fitting effect is better. Finally, we throw the trained model into a regression model to calculate the humor rate and controversial rate.

## 4 Experimental setup

The code this time is mainly based on [Transformers](#) under Hugging Face. The neural network tool we use is PyTorch. For sub-task 1a and sub-task 1c, we use the same method A (the same structure). We just read different id of training set. Sub-task 1b and sub-task 2 used the other method B.

### 4.1 Hyper-parameters

In this work, because our models are implemented based on PyTorch. We use the BERT-base+CNN as our sub-task 1a and sub-task 1c’s pre-trained model. For all models, in order to save GPU memory, the batch size parameter of GPU in fine-tuning is set to 8 and the gradient accumulation steps (gas) is set to 1, so that each time a sample is an input, the gradient is accumulated 1 times, and then the back-propagation update parameters are performed. The memory is saved by sacrificing a certain training speed; learning rate is  $5e-5$ . we use the triangular learning rate. First, the learning rate is gradually increased through warm up, and then the linear learning rate is gradually reduced through linear learn rete decay, which effectively improves the training effect. (Bai and Zhou, 2020) The hyper-parameters of each model and the results on the test set are shown in Table 1.

Tasks	Hyperparameters
Sub-task 1a and sub-task 1c	lr= $5e-5$ output hidden states=True epoches=5 per gpu train batch size=8 gas=1 filt size=(3,4,5) num filter=(3,4,5) hidden size=68 dropout=0.2
Sub-task 1b and sub-task 2	output hidden states =True dropout=0.1 lr= $5e-5$ epoches=10 per gpu train batch size=8 gas=1

Table 1: Hyperparameters of the used model

### 4.2 Prediction module

For sub-task 1a and sub-task 1c, we mainly use Precision and F1-score to evaluate our model A. The

criteria evaluation of F1-score is as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{Precision * Recall * 2}{Precision + Recall}$$

For task 1b and task 2, we use RMSE to evaluate our model. RMSE is as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (y_i - \hat{y}_i)^2}{N}}$$

## 5 Results

For sub-task 1a and sub-task 1c, we first tried [BERT-base](#) as our word embedding model. On this basis, we added LSTM (Wang et al., 2018) to further extract the features of words. Long-term short-term memory (LSTM) network has the ability to maintain long-term memory. The ability of has proven to be particularly useful for learning sequences containing long-term patterns of unknown length. We also tried two other pre-training models (ALBERT and RoBERTa) as a comparative experiment. These two new language models have made some improvements on the basis of BERT, but they have different effects on different data sets. RoBERTa is suitable for complex neural network architecture, ALBERT architecture is relatively streamlined. From the data in Table 2, it can be seen that they are almost the same as BERT in extracting word features. Finally, we chose to add CNN on the basis of BERT to extract the features of words, and found that the effect is better than LSTM and other pre-training models.

Since CNN is not very effective in dealing with regression problems, we mainly use RoBERTa as our system architecture. From Table 3, we can find that compared to BERT, RoBERTa’s RMSE is far better than BERT-base under the same training epochs.

## 6 Conclusion

In this paper, we gave a description of the BERT+CNN architecture and the popular RoBERTa pre-training model architecture, and finally solved four sub-tasks. The best F1 for Sub-task 1a is 0.9206 and the best F1 for Sub-task

1c is 0.6744. The best root mean square errors of Sub-task 1b and Sub-task 2 are 0.6510 and 0.5588. Our four sub-tasks all appeared on the leaderboard (Note: The data from our computer is a little different from the official evaluation results. The results of the final four sub-tasks in the leadboard are shown in Table 4.). Experiments have shown that CNN has a certain effect on text classification, but this time only one layer of CNN was added to the Classifier. In addition, the experiment also shows that RoBERTa has a better effect than BERT in dealing with regression problems. In the future, we will try to integrate and distill the model and process the data. We consider to introduce external knowledge to model headlines and improve the humor recognition performance.

Method	Task 1a		Task 1c	
	F1	Acc	F1	Acc
BERT	0.9175	0.9310	0.6659	0.6835
ALBERT	0.9162	0.9210	0.6576	0.6911
RoBERTa	0.9176	0.9320	0.6488	0.7300
BERT+LSTM	0.9054	0.9120	0.6519	0.6806
BERT+CNN	<b>0.9206</b>	0.9250	<b>0.6744</b>	0.7025

Table 2: the Table for Sub-task 1a and Sub-task 1c

Method	Sub-task 1b	Sub-task 2
	RMSE	RMSE
BERT	1.7161	1.826
ALBERT	0.6700	0.6576
RoBERTa	<b>0.6510</b>	<b>0.5588</b>

Table 3: the Table for Sub-task 1b and Sub-task 2

Task	Best Result
sub-task 1a	0.9205(F1)
sub-task 1b	0.7010(RMSE)
sub-task 1c	0.4271(F1)
sub-task 2	0.5419(RMSE)

Table 4: Final Result on the Leaderboard

## 7 Acknowledgments

Our work was supported by the Natural Science Foundations of China under Grants 61463050, the NSF of Yunnan Province under Grant 2015FB113.

## References

- Charlotte Ammer and Lea Grüner. 2020. [UniTuebingenCL at SemEval-2020 task 7: Humor detection in news headlines](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1060–1065, Barcelona (online). International Committee for Computational Linguistics.
- Yang Bai and Xiaobing Zhou. 2020. [BYteam at SemEval-2020 task 5: Detecting counterfactual statements with BERT and ensembles](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 640–644, Barcelona (online). International Committee for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and Dario Amodei. 2020. Language models are few-shot learners.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). *CoRR*, abs/1810.04805.
- Anna-Katharina Dick, Charlotte Weirich, and Alla Kutkina. 2020. [HumorAAC at SemEval-2020 task 7: Assessing the funniness of edited news headlines through regression and trump mentions](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1019–1025, Barcelona (online). International Committee for Computational Linguistics.
- Yoon Kim. 2014. [Convolutional neural networks for sentence classification](#). *CoRR*, abs/1408.5882.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. [ALBERT: A lite BERT for self-supervised learning of language representations](#). *CoRR*, abs/1909.11942.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th*

*Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing.*

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*.

Ali Safaya, Moutasem Abdullatif, and Deniz Yuret. 2020. [KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2054–2059, Barcelona (online). International Committee for Computational Linguistics.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. [How to fine-tune BERT for text classification?](#)

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.

Jin Wang, Bo Peng, and Xuejie Zhang. 2018. Using a stacked residual LSTM model for sentiment intensity prediction. *Neurocomputing*, 322(DEC.17):93–101.

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *NIPS (2017)*.

# DLJUST at SemEval-2021 Task 7: Hahackathon: Linking Humor and Offense

**Hani Al-Omari**

Jordan University of Science  
and Technology  
Computer Information Systems  
Department  
Irbid, Jordan  
alomarihani1997@gmail.com

**Isra'a AbedulNabi**

Jordan University of Science  
and Technology  
Computer Information Systems  
Department  
Irbid, Jordan  
israha95@gmail.com

**Rehab Duwairi**

Jordan University of Science  
and Technology  
Computer Information Systems  
Department  
Irbid, Jordan  
rehab@just.edu.jo

## Abstract

Humor detection and rating poses interesting linguistic challenges to NLP; it is highly subjective depending on the perceptions of a joke and the context in which it is used. This paper utilizes and compares transformers models; BERT base and Large, BERTweet, RoBERTa base and Large, and RoBERTa base irony, for detecting and rating humor and offense. The proposed models, where given a text in cased and uncased type obtained from SemEval-2021 Task7: HaHackathon: Linking Humor and Offense Across Different Age Groups. The highest scored model for the first subtask: Humor Detection, is BERTweet base cased model with 0.9540 F1-score, for the second subtask: Average Humor Rating Score, it is BERT Large cased with the minimum RMSE of 0.5555, for the fourth subtask: Average Offensiveness Rating Score, it is BERTweet base cased model with minimum RMSE of 0.4822.

## 1 Introduction

SemEval 2021 Task7 is constructed to detect and rate the humor and offense inside jokes in the English language (Meaney et al., 2021). Humor is an essential aspect of strengthening human communication and relations. However, the interpretation of humor differs based on the perceptions of a joke and the context in which it is used. In 2012, the Human Rights Commission found the most commonly reported form of harassment in Australia was sexist or offensive jokes (the, 2012), humor appreciation; is a highly subjective phenomenon as a sense of humor varies from person to person depending on factors such as age, gender, and socio-economic status. In this task, data labels and ratings were collected from a balanced set of age groups from 18-70. Moreover, annotators represent a variety of genders, political stances, and income levels. The automatic detection of linguistic elements

in natural language texts such as aggression, humor, irony, and sarcasm has drawn attention to research communities (Davidov et al., 2010). Several studies and experiments have been performed to develop and improve humor detection systems (Annamoradnejad, 2021) (Winters and Delobelle, 2020) (Sane et al., 2019) (Mao and Liu, 2019) (Chen and Soo, 2018). BERT language model (Annamoradnejad, 2021) (Devlin et al., 2019) showcase the highest results compared to all other works. This paper aims to utilize the Transformers models; BERT base and Large (Devlin et al., 2019), BERTweet (Nguyen et al., 2020), RoBERTa base and Large (Liu et al., 2019), and RoBERTa base irony (Barbieri et al., 2020) for humor detection, humor rating, and offense rating using the dataset obtained from SemEval-2021 Task7 that contains training, development, and test data. Our contributions are: Preprocessing text techniques for text tokenization, word segmentation, spell correction, removing the punctuation, encoding, and extracting embeddings. Furthermore, training six state-of-the-art Transformers Models and compare its results against the Base-line Model. We have achieved a 0.9540 F1-score for Subtask1-A Humor Detection using BERTweet cased model compared to the first place score, which is 0.9820 F1-score. For Subtask1-B Average Humor Score, our RMSE result is 0.5555 using BERT Large cased model, the first place RMSE is 0.4959. Finally, for Subtask2 Average Offensiveness Score, our RMSE results is 0.4822 using BERTweet cased model, while the first place RMSE is 0.4120.

This paper's structure is as follows: Section 2 reviews related works focused on Humor detection in SemEval-2020. Section 3 presents data exploration, preprocessing, training models, and evaluation metrics. Section 4 introduces the experiments and results. Section 5 remarks the conclusion and proposes future works.

## 2 Related Work

The previous Humor Detection task on SemEval2020 Task7 was focused on humor rating only without considering if it is offensive or not. However, in SemEval2021 Task7, the task requires detecting the hidden offense inside jokes. Rozen et al., 2020 (Rozen et al., 2020) presented a novel L2-Regularization approach with freezing the weights for the first epoch to train and fine-tune the word embedding model, ensemble different language models - BERT, XL-NET, and Roberta, and duplication from each language models, with a weighted average between them. Their approach ranked second place in SemEval-2020 Task 7: "Assessing Humor in Edited News Headlines", subtasks 1 and 2.

Shatnawi et al., 2020 (Shatnawi et al., 2020) also proposed the BERT-Flair-based Humor Detection Model (BFHumor) that combined the BERT regressor and Flair library to predict the funniest values of edited headlines for the same Task 7 of SemEval 2020; the mode ranked 4th in subtask1 and 12th in the subtask2. Meanwhile, Pramodith Ballapuram 2020 (Ballapuram, 2020) participated in the same task using a non-ensemble model; he proposed a Siamese Transformer based approach, coupled with an Attention mechanism to make use of contextual embeddings and focus words and their impact against other tokens on generating important features and rating the funniness of the edited headline, he scored fifth place in subtask1 and fourth place in subtask2.

## 3 Methodology

Our methodology of tackling the humor detection problem consists of four phases: Data exploration and Visualization, Data Pre-Processing, Learning Models, and Evaluation Criteria.

### 3.1 Data Exploration

The dataset from SemEval-2021 Task7: Ha-Hackathon: Detecting and Rating Humor and Offense, consists of five columns as table 1 shows; col-1 is the id of the text, col-2 "text" is the raw text for a joke to process, col-3 "is-humor" is a binary classification for the text, 1 means it is humor and 0 means it is not humor, col-4 "humor-rating" is a numerical representation for how much humorous is the text if it is labeled as humor from col-2, col-5 "humor-controversy" is binary classification to represent the subjectivity of humor appreciation

with a controversy score, 1 means the humor of the text is controversial and 0 is not, col-6 "offense-rating" is a numerical representation for how much offensive is the text. The competition consists of two subtasks: subtask1 is divided into three parts A,B and C and predicts is-humor, humor-rating, and humor-controversy respectively. Subtask2 is to predict the offense-rating.

Data sections are described separately, starting with subtask1-A; it is a binary classification problem to detect whether the text is humor or not. The distribution between its classes is balanced, so no need for data upsampling or downsampling. Subtask1-B is dependent on subtask1-A; if the text is labeled as humorous, a value will be provided to humor rating, and if it is not, the rating will be none. For this task, we dropped records that are labeled as not humorous. Humor rating is a regression problem since the rating is a continuous value between zero and five, the values are distributed normally which helps the model to generalize better. Subtask1-C also depends on subtask1-A; if the text is classified as humorous, then predict if the humor rating would be considered controversial. It is a binary classification task, and the distribution between its classes is balanced. Table 2 shows the number of instances per each class for both "is\_humor" and "humor\_controversy". Subtask2 is a regression problem to predict how offensive a text would be, the target value is between zero and five. After checking the target distribution, it is skewed to the left, which indicates the model will have difficulty in training, reducing the chances of predicting the values above 3.

We explored the number of the words distribution per instance and the number of unique words; where we do not count the same words. Both distributions were similar in density, that indicated most of the text content are unique and non-repetitive words. the maximum number of words within instances is 70 words for the whole data set and the average number of words is around 20 words. We choose to define the input sequence length equal to 128 as through tokenization, some words will be divided to multiple tokens. To check odd or very long words that need to be handled, we visualize the mean word length for each instance in the data set, and the word length is in the average mean. Finally, we checked the number of punctuations in the text since it affects the model as it affects the text in the encoding phase, especially when it is

id	text	is_humor	humor_rating	humor_controversy	offense_rating
1	TENNESSEE: We're the best state. Nobody even comes close. *Elevnessee walks into the room* TENNESSEE: Oh shit...	1	2.42	1	0.2
297	I met a vaping vampire from Romania. He called himself Vlad the Inhaler.	1	2.05	0	0
4698	What's the difference between black people and cancer? Cancer got Jobs.	1	1.75	0	4.2
5231	Fellas: Don't be mad when someone else starts to appreciate the woman you took for granted. What you won't do, someone else will.	0			0.3
5300	Black people love boom boxes .. I hate to generalize, but it's their stereotype :-)	1	1.54	0	2.9

Table 1: Train dataset

feature name	negative	positive
is_humor	3068	4932
humor_controversy	2467	2465

Table 2: Number of instances that belong to each class for "is\_humor" and "humor\_controversy"

attached with a word (e.g., animals do not encode the same as animal's). We handle the punctuation in the preprocessing part, which will be described in the next section.

### 3.2 Data Pre-Processing

In this phase, we apply enhancement techniques to the text. Removing duplicate sentences, repetitive characters, spilling mistakes, and stop words. Also, it includes encoding methodology to transform text from its original form to a vector that makes the computer understands it. We used Ekphrasis (Baziotis et al., 2017) for spell correction, remove contraction words, and annotate caps text as it is crucial to know the speaker's tone, capital letters indicate a level of aggression. We used GloVe (Pennington et al., 2014) vocabulary to find the out of vocabulary words that Ekphrasis did not fix, applied some spell correction manually, and removed the punctuation that has been attached to some words using regular expressions. We applied the tokenization technique. For GloVe embeddings, using Keras tokenization tool that splits the tokens based on the space. For example ["We went to Aqaba."] will be tokenized as the following: ['We', 'went', 'to', 'Aqaba.'], and each token gets encoded. On the other hand, the BERT model uses a WordPiece tokenizer that depends on its own vocabulary, and if it faces an out-of-vocabulary word, it will be split into sub tokens that starts with ##token. For example, ['tokenization'] become ['token', '##ization'], and for the RoBERTa model, it uses byte pair encoding (BPE) word pieces. RoBERTa handles the out-of-vocabulary words the same way as the BERT model but with some modification on the algorithm. For example: ['tokenization'] become ['token', ' ization']. We encoded the text using GloVe and transformers; Glove considers frequency when building

the embeddings, unlike word2vec.

### 3.3 Learning Models

This section will describe the models we have used; it will be divided into two sub-sections: Baseline Model, and Transformers Models.

#### 3.3.1 Base Line Model

The baseline model is BiLSTM model. It takes an encoded sentence with 100 token sequence size as input, each encoded using GloVe embeddings that have been fed into the embedding layer; which considered to be as a lookup table which consists of 300-dimensional pretrained GloVe embeddings, and each row in the table is considered a representation of the word. Next is two BiLSTM models that consist of 128 nodes, 0.2 dropouts to avoid overfitting, and He uniforms weight initializer (He et al., 2015). Output passes through a feed-forward network which consist of four hidden layers of 512, 256, 128 and 64 neurons respectively, for each layer we use ReLU activation function, 0.4 dropouts, and He uniforms weight initializer. Finally, the output layer consists of the Sigmoid activation function in binary classification and linear layer for the Regression task. We used Stochastic Gradient Descent (SGD) with a 0.01 learning rate, 0.99 momentum, and Nesterov implementation (Nesterov, 2003). It is worth mentioning that we used the early stopping technique to avoid overfitting with five patience.

#### 3.3.2 Transformers Models

We have applied different type of pretrained models using Simple Transformers library (Rajapakse) which is an API built above Hugging Face library (Wolf et al., 2019). In this section, we generally describe the models that we used. Bidirectional Encoder Representation from Transformers (BERT) is a pretrained model which uses attention models to learn the contextual relation between the words in the sentence, consisting of two main parts: an encoder and a decoder: an encoder that encodes the text, and a decoder for the output result based on the task. We used Bert cased and uncased models, which both have been trained on BookCoupus (Zhu et al., 2015) with 800 million words and En-



English Wikipedia with 2,500 million words. We trained the model using this hyperparameter: 128 sequence length, three epochs, 32 batch size, 4e-5 learning rate, and AdamW as an optimizer. Then we used the BERTweet model trained on BERT architecture using 850 million English tweets. We trained this model using almost the same hyperparameters, except that we used eight as batch size. We have used, too, Robustly Optimized BERT pre-trained approach (RoBERTa). It is a fine-tuned version of the BERT model with some changes on the data size and input representation. Training the model on larger data set significantly improves the model performance using BookCorpus, English Wikipedia, CC-news with 63 million English news articles, OpenWebText, and Stories; which is a subset of CommonCrawl data. Model developers use dynamic masking instead of static masking that has been used in the BERT model, this technique allows to improve the model performance. Furthermore, they have used the full sentence without using the next sentence prediction loss. We trained both RoBERTa base and large models using the following hyperparameters: for the base three 128 sequence length, three epochs, 32 batch size, 4e-5 learning rate, and AdamW optimizer, and the large model, we kept everything the same as the base model but change the batch size to 16. Moreover, we have used the RoBERTa irony model that has been fine-tuned using 58 million tweets for irony detection on TweetEval benchmark; also, we used everything as the base and large model. We have also used XLM-RoBERTa (Conneau et al., 2020), and XLNet (Yang et al., 2019) as a black box, we did not go into the model’s detail, but in general, it is an improved version model from BERT. We applied them using the following hyperparameters: 128 sequence length, three epochs, 16 batch size, 4e-5 learning rate, and AdamW optimizer.

### 3.4 Evaluation Criteria

F1 score criteria was used for the binary classification task and the second criteria is RMSE for the regression task. We used the 8-Fold Cross-validation method to determine the best model since we only have an 8k data instances for training. We trained our models using seven folds, kept the last fold unseen for validation in each iteration, and used every model from each iteration to predict the development and testing data and combine all the results to obtain the final result.

## 4 Experimentation and Results

### 4.0.1 Task 1-A Is-Humor:

We constructed a baseline model and fine-tuned it using different approaches and preprocessing techniques. We tested our model using four types of preprocessing techniques (None, Ekphrasis, Ekphrasis with removing stop words, and Ekphrasis with applying Custom Spell Correction). After that, we tested Adam and SGD as optimizers, SGD performs better on this model. After comparison, the best parameters for the models are described in the fifth row of table 3; we applied the early stopping technique to reduce the overfitting with 0.4 dropout. We test Transformers models, refer to the table 4. In general, we fine-tuned all the models in two ways, the Cased model, which means that the text is kept in its original form without lowering the characters’ case. Uncased means that lower-case all the characters in the text. By experiment, cased models performed better than uncased since it captures more aggressive behaviors from the writer. Moreover, BERTweet performs well using the Cased model on this task. Since the model has already been trained on Twitter data, it captures all the slang, acronyms, and abbreviations.

### 4.0.2 Task 1-B: Humor Rating

We used the experiments from the previous task since they are dependent if humor equals zero; we do not need to predict the humor rating. If it is one, we have to predict the humor rating value between zero and five. We first used the best model from the baseline by changing the last layer to a linear layer to predict continuous values since it is a regression task. Same hyperparameters from model 5 from table 3 we got 0.8651 RMSE, and we consider it as our baseline. After that, we tested the best transformer models from the previous task, refer to table 5. We found out that the best model is the BERT large case model, which seems unexpected since most of the other models perform almost the same, around 0.54.

### 4.0.3 Task 2 Offensiveness Score:

We have used the same methodology from the Subtask1-B; we first constructed a baseline for this task, which is the best model from the Subtask 1-A hyperparameters model 5 from table 3, and we got 0.86783 RMSE. After that, we tested the best models from the previous task’s transformers, refer to table 6. We found out that the best model is BERT

	Pre-Processing	learning rate	# epochs	dropout	batch size	optimizer	features	Accuracy	Precision	Recall	F1-score
1	None	0.0001	50	.4	128	Adam	GloVe	0.8594	0.8778	0.8968	0.8872
2	Ekph + remove stopwords	0.0001	50	.4	128	Adam	GloVe	0.8535	0.8804	0.8821	0.8813
3	Ekph	0.0001	50	.4	128	Adam	GloVe	0.875	0.9083	0.8867	0.8974
4	Ekph + Custom Spell-Correction	0.0001	50	.4	128	Adam	Glove	.88062	0.9066	.8990	0.9028
5	Ekph + Custom Spell-Correction	0.0001	50	.4	128	SGD	Glove	0.8806	0.9003	0.9067	0.9035

Table 3: Baseline model experiments on is-humor task

	Model	Text Type	# epochs	batch size	Accuracy	Precision	Recall	F1-score
1	BERT base uncased	Uncased	3	32	0.9424	0.9515	0.9552	0.9533
2	BERT Large uncased	Uncased	3	32	0.9455	0.9582	0.9532	0.9556
<b>3</b>	<b>BERTweet uncased</b>	<b>Uncased</b>	<b>3</b>	<b>8</b>	<b>0.9561</b>	<b>0.9679</b>	<b>0.9607</b>	<b>0.9643</b>
4	RoBERTa base	Uncased	3	32	0.9448	0.9593	0.9548	0.9570
5	RoBERTa Large	Uncased	3	16	0.8366	0.8366	0.9686	0.8978
6	RoBERTa base irony	Uncased	3	32	0.9494	0.9574	0.9607	0.9590
7	XLNet-RoBERTa large	Uncased	3	16	0.7408	0.7087	0.9837	0.8239
8	XLNet base	Uncased	3	16	0.9449	0.9572	0.9531	0.9551
9	XLNet large	Uncased	3	16	0.8030	0.7750	0.9588	0.8571
10	BERT base cased	Cased	3	32	0.9440	0.9558	0.9532	0.9545
11	BERT Large cased	Cased	3	32	0.9505	0.9597	0.9600	0.9599
<b>12</b>	<b>BERTweet cased</b>	<b>Cased</b>	<b>3</b>	<b>32</b>	<b>0.9589</b>	<b>0.9704</b>	<b>0.9627</b>	<b>0.9665</b>
13	RoBERTa base	Cased	3	32	0.9494	0.9615	0.95612	0.9588
<b>14</b>	<b>RoBERTa Large</b>	<b>Cased</b>	<b>3</b>	<b>32</b>	<b>0.9563</b>	<b>0.9702</b>	<b>0.9584</b>	<b>0.9643</b>
15	RoBERTa base irony	Cased	3	16	0.9543	0.9633	0.9625	0.9629

Table 4: Transformers models on is-humor task

Model	Text Type	# epochs	batchsize	RMSE
BERTweet	Uncased	3	8	0.5479
RoBERTa base	Uncased	3	32	0.5417
BERT base	Cased	3	32	0.5360
<b>BERT Large</b>	<b>Cased</b>	<b>3</b>	<b>32</b>	<b>0.5296</b>
BERTweet	Cased	3	32	0.54585
RoBERTa base	Cased	3	32	0.54272
RoBERTa Large	Cased	3	32	0.54548
RoBERTa irony	Cased	3	16	0.54585

Table 5: Transformers models on humor rating

Large model and it performs well on regression tasks since it performs well on the two tasks related to regression. All the previous experiments applied the cross-validation method on the training data. Since we decided that we do not want to overfit the development and test dataset, we will use the best models that perform well on the cross-validation phase to predict the development and test dataset’s output. We changed the threshold point for the is-humor task; using the ROC curve, and the best split is 0.233357 since it improves the model by 0.04 percent on the development phase, so we apply it to the testing phase. The best model for the is humor task is BERTweet cased base model that scored a 0.9540 F1 score on the testing phase; for the humor rating task, we used BERT large cased model that scored 0.5555 RMSE on the testing phase, and for

the offensive score, we used BERTweet large cased model that scored 0.4822 RMSE on testing phase.

Model	Text Type	# epochs	batchsize	RMSE
BERTweet	Uncased	3	8	0.5304
RoBERTa base	Uncased	3	32	0.5734
RoBERTa Large	Uncased	3	32	0.5494
BERT base	Cased	3	32	0.5516
<b>BERT Large</b>	<b>Cased</b>	<b>3</b>	<b>32</b>	<b>0.5247</b>
BERTweet	Cased	3	32	0.5302
RoBERTa base	Cased	3	32	0.5522
RoBERTa Large	Cased	3	32	0.7190
RoBERTa irony	Cased	3	16	0.8501

Table 6: Transformers models on offensiveness score

## 5 Conclusion and Future Work

In this paper, we experimented with a set of state-of-the-art Transformers and contextual models for detecting and rating humor and offense in text. Our experimental results show that the BERTweet Large model is the best model for humor binary classification task with a 0.9540 F1 score and offensive rating with 0.4822 RMSE, and BERT Large cased model is the best for humor rating task scored 0.5555 RMSE. We plan to enhance the top models using ensemble learning methodology and test out more novel methods.

## References

2012. Working without fear: Results of the sexual harassment national telephone survey.
- Issa Annamoradnejad. 2021. Colbert: Using bert sentence embedding for humor detection.
- Pramodith Ballapuram. 2020. Lmml at semeval-2020 task 7: Siamese transformers for rating humor in edited news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1026–1032.
- Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. 2020. TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*.
- Christos Baziotis, Nikos Pelekis, and Christos Doukolidis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.
- Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117.
- Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale.
- Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.
- Jihang Mao and Wanli Liu. 2019. A bert-based approach for automatic humor detection and scoring. In *IberLEF@ SEPLN*, pages 197–202.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Yurii Nesterov. 2003. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Thilina Rajapakse. [link].
- Alon Rozenal, Dadi Biton, and Ido Blank. 2020. Amobee at semeval-2020 task 7: Regularization of language model based classifiers. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 981–985.
- Sushmitha Reddy Sane, Suraj Tripathi, Koushik Reddy Sane, and Radhika Mamidi. 2019. Deep learning techniques for humor detection in hindi-english code-mixed tweets. In *Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 57–61.
- Fara Shatnawi, Malak Abdullah, and Mahmoud Hammad. 2020. Mlengineer at semeval-2020 task 7: Bert-flair based humor detection model (bfhumor). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1041–1048.
- Thomas Winters and Pieter Delobelle. 2020. Dutch humor detection by generating negative examples.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.
- Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

# Gulu at SemEval-2021 Task 7: Detecting and Rating Humor and Offense

**Maoqin Yang**

School of Information, Yunnan University, Yunnan, P.R. China

maomaoq33@gmail.com

## Abstract

Humor recognition is a challenging task in natural language processing. This document presents my approaches to detect and rate humor and offense from the given English text. This task includes 2 tasks: task 1 which contains 3 subtasks (1a, 1b, and 1c), and task 2. Subtask 1a and 1c can be regarded as classification problems and take ALBERT as the basic model. Subtask 1b and 2 can be viewed as regression issues and take RoBERTa as the basic model. And finally, team-Gulu scores in subtask 1a with a weighted average F1 score of 0.9190, in subtask 1b with an RMSE score of 0.7405, in subtask 1c with a weighted average F1 score of 0.5561, and in subtask 2 with an RMSE score of 0.5807 on the private leader board.

## 1 Introduction

For social animals like humans, humor is an effective bonus. From the perspective of evolutionary psychology, “humorous” often means superior creativity, in other words, a smarter mind. Therefore, it is important to recognize whether a sentence is humorous and how humorous the sentence is. It is a bit impractical to recognize such a huge data set by humans, so it becomes necessary for us to develop a system to automatically detect humor. In this task, the organizer collects labels and ratings from a balanced age group of 18-70. Annotators also represent various genders, political positions, and income levels. Therefore, for some texts classified as humorous, we should once again prove whether they are controversial and predict the offensiveness of the text. For more specific content, please refer to the official website of the competition<sup>1</sup>.

<sup>1</sup><https://competitions.codalab.org/competitions/27446>

Because the pre-trained and deep learning models have shown excellent performance in many NLP problems such as classification and topic extraction(Zampieri et al., 2019), so I use deep learning methods to deal with those four tasks. According to the latest related research progress, the transformer-based language model has become my favorite model. In order to facilitate the understanding of the corresponding model of each subtask, I made it into a table shown as Table 1. I choose A Lite BERT (ALBERT)(Lan et al., 2019) as my basic model in subtask 1a. In subtask 1b and subtask 2, I choose Bidirectional Encoder Representations for Transformers (BERT)(Devlin et al., 2018) model as my basic model. In subtask 1c, A Robustly Optimized BERT (RoBERTa)(Liu et al., 2019) has been chosen. To get a more effective and higher accuracy model in subtask 1a, BiGRU combined with attention. To prove the effectiveness of this model, there are also comparative experiments with other neural networks for task 1c. To obtain as much effective information as possible from the limited data, the 5-fold cross-validation method has been used.

## 2 Related Work

Automatic humor recognition is a very challenging research topic in natural language processing. A person’s degree of humor is largely determined by his educational knowledge and common sense of life. In addition, many types of humor require a lot of external knowledge, such as irony, metaphor and satire.

Yang et al. (2015) first determined the semantic structure behind each structure of the humor and design feature set, and then used a calculation method to identify the humor and their humor recognizer was very effective in automatic distinction humorous and non-humorous text.

subtask	1a	1b	1c	2
category	classification	reg	classification	reg
model	ALBERT+BiGRU	BERT	RoBERTa+BiLSTM+BiGRU	BERT

Table 1: The category and model used for each subtask, where the *reg* stands for *regression*

Morales and Zhai (2017) proposed a generative language model based on the inconsistency theory to model humorous text, so that they can use background text sources such as Wikipedia item descriptions, and can build multiple functions for recognizing humorous comments. Using supervised learning to classify reviews into humorous reviews and non-humorous reviews, these functions showed that the features constructed based on the proposed generative model were more effective than the main features proposed in the existing literature.

Liu et al. (2018) found that certain grammatical structural features are consistently related to humor. Both experimental results and analysis showed that humor can be regarded as a style, and the content-independent syntactic structure can help identify humor and had good explanatory power. Therefore, they proposed to use syntactic structure features to enhance humor recognition ability. Compared with the baseline driven by humor theory, their method had achieved a significant improvement.

And subtask 1b and 2 are regression problems. We need to predict the humor of a sentence, and because the implicit meaning of the sentence may be offensive for someone, we also need to detect the degree of attack on each sentence. Traditionally, text regression is solved using linear models. Bitvai and Cohn (2013) proposed a method based on a deep convolutional neural network (CNN). Yang et al. (2015) recommended using copula: a powerful statistical framework. Their model clearly outperformed a strong linear and nonlinear discrimination baseline. Subramanian et al. (2018) used CNN regression with auxiliary ordinal regression objective to predict the popularity of petitions in their work. A regression task is actually a special form of the classification task. The final output is a value rather than the probability of a specific category. Therefore, the BERT model can achieve good results.

### 3 Materials and Methods

#### 3.1 Preprocessing

The given data (Meaney et al., 2021) contains the tasks required by each subtask, but some corresponding data are missing. For example, if a sentence is judged as not humorous in 1a, there will be no data in the column (the fourth and fifth column) corresponding to subtask 1b and 1c. To facilitate the experiment, I added 0 to all missing values.

#### 3.2 Data set

Given a sentence, for 1a, the system must assign the label to 1 if it is recognized as *is\_humor*, otherwise, assign it to 0. And if there is a *humor\_controversy* in 1c, the corresponding label is 1. For this task, the available sentences including 6948 training sentences, 1052 development sentences, and 1000 testing sentences. The label distribution in the training set is almost balanced for 1c, but for 1a, label 1 accounts for only 38.1% of the total after assigning all missing values to 0. The number of sentences for each label is listed in Table 2.

task	label 0	label 1
subtask 1a	2652	4296
subtask 1c	2149	2147

Table 2: The distribution of training set

I divide all the data of subtask 1b and 2 into 5 intervals (take 1 as the step size) and count the total of each interval. It can be seen from Figure 1 that most of the data of *humor\_rating* are between 1 and 2 (1361 sentences), and there is no data between 4 and 5. The label *offense\_rating* scores of 0 accounted for the majority (2913 data in total).

#### 3.3 Classification

Text classification is the most basic and very necessary task in natural language processing (NLP). Two of this task belongs to classification problems. In subtask 1a, I combine ALBERT with BiGRU-Attention. In subtask 1c, I combined RoBERTa with BiLSTM+BiGRU.

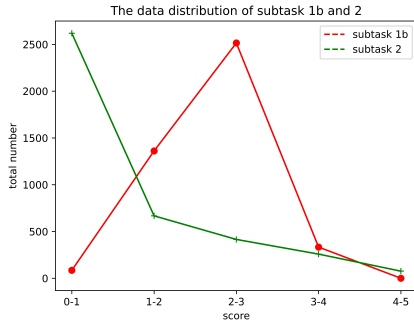


Figure 1: The data distribution of subtask 1b and 2

### 3.3.1 ALBERT + BiGRU-Attention

The ALBERT model is an improvement based on the BERT model. The ALBERT model<sup>2</sup> has designed parameter reduction by changing the result of the original embedding parameter  $P$  (the product of the vocabulary size  $V$  and the hidden layer size  $H$ ).

$$V * H = P \rightarrow V * E + E * H = P \quad (1)$$

Where  $E$  represents the size of the low-dimensional embedding space. In ALBERT,  $H \gg E$ . The self-supervised loss is used to focus on the internal coherence in the construction of sentences<sup>3</sup>.

The BiGRU-Attention model<sup>4</sup> is divided into three parts: text vector input layer, hidden layer, and output layer. Among them, the hidden layer consists of three layers: the BiGRU layer, the attention layer, and the Dense layer (fully connected layer). The output of the ALBERT model will be used as the input. After receiving the input, the BiGRU neural network layer will extract features of the deep-level information of the text firstly. Secondly, it uses the attention layer to assign corresponding weights to the deep-level information of the extracted text. Finally, the text feature information with different weights is put into the softmax function layer for classification.

In order to improve the classification ability of the model, I combined ALBERT and BiGRU-Attention. The model diagram is shown in Figure 2.

<sup>2</sup><https://huggingface.co/albert-base-v2>

<sup>3</sup><https://zhuanlan.zhihu.com/p/162275803>

<sup>4</sup>[https://blog.csdn.net/qq\\_40900196/article/details/88998290](https://blog.csdn.net/qq_40900196/article/details/88998290)

### 3.3.2 RoBERTa + BiLSTM + BiGRU

RoBERTa<sup>5</sup> mainly made several adjustments based on BERT: 1) Longer training time, larger batch size, more training data; 2) Removed next predict loss; 3) Longer training sequence; 4) Dynamic adjustment Masking mechanism.

Using the BiLSTM model can better capture the two-way semantic dependence. Because LSTM can learn what information to remember and what information to forget during the training process. BiGRU is a unidirectional, opposite direction, and outputs a neural network model composed of GRUs determined by the states of these two GRUs. At each moment, the input will provide two GRUs in opposite directions at the same time, and the output will be jointly determined by the two unidirectional GRUs.

In order to improve the ability of the model, I combined RoBERTa and BiLSTM+BiGRU. The model diagram is shown in Figure 3.

### 3.4 Regression

What regression predictive modeling needs to accomplish is to approximate a mapping function from an input variable to a continuous output variable. Both regression subtasks use the BERT model.

The BERT model implements three embedding layers: position embedding, word embedding, and segment embedding. BERT uses two training strategies: the masked language model and the next sentence prediction. The language model trained in this way usually has a deeper sense of language context and can be further applied to process various NLP tasks( classification et.), with an additional output layer(Fan et al., 2019).

### 3.5 Evaluation

The main metric for the classification tasks will be f1-measure(wei et al., 2020).

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \quad (2)$$

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \quad (3)$$

$$F1 = \frac{2 * precision * recall}{precision + recall} \quad (4)$$

<sup>5</sup><https://huggingface.co/roberta-base>

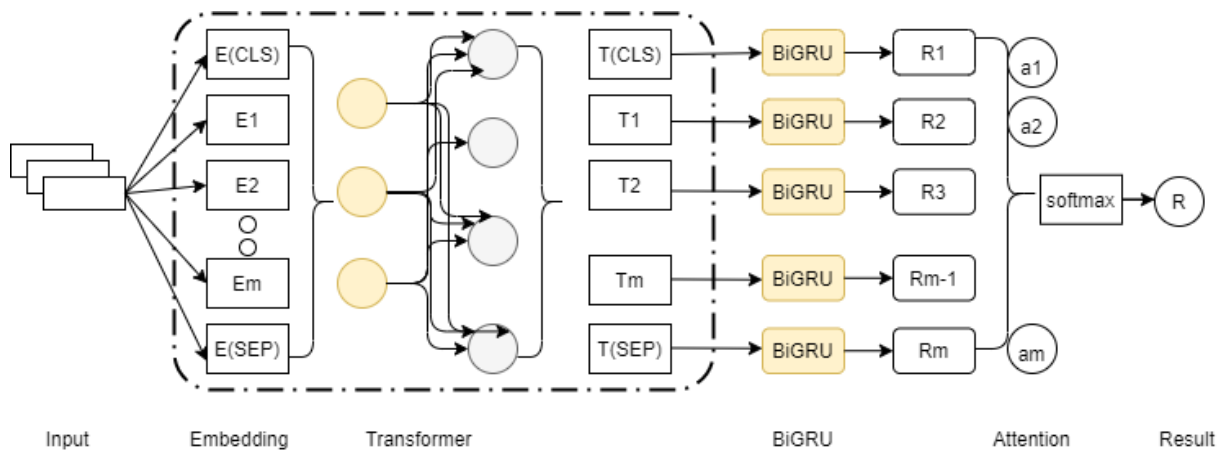


Figure 2: ALBERT+BiGRU-Attention for task 1a, where the  $E[CLS]$  and  $E[SEP]$  are added at the beginning and end of each instance respectively

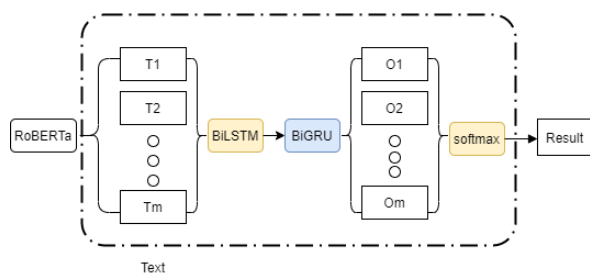


Figure 3: The model for task 1c

Model	step	batch size	lr	epoch
BERT	500	32	2e-5	2
ALBERT	2500	32	2e-5	10
RoBERTa	5000	16	2e-5	10

Table 3: The parameters, where the  $lr$  stands for *learningrate*

The metric for the regression tasks will be the root mean squared error (RMSE). Below  $x$  and  $y$  are  $D$  dimensional vectors, and  $x_i$  represents the value of  $x$  in the  $i$ th dimension.

$$RMSE = \sqrt{\sum_{i=1}^D (x_i - y_i)^2} \quad (5)$$

## 4 Results

In this task, I used ALBERT, RoBERTa, and BERT models for the training task. For these models, the main hyperparameters I want to pay attention to are the training step size, batch size learning rate, and epoch. The parameters of my model are shown in Table 3.

### 4.1 Classification results

For task 1c, several sets of comparative experiments were carried out. The comparison results are listed in Table 4, and the cross-validation results are 0.92, 0.94, 0.94, 0.93, and 0.67 respectively.

All results are the results of the evaluation set. The output of the classification result is shown in Table 5. We can see that the number of label 1 is close to twice the number of label 0.

Task 1a scores 0.9190 but 1c scores 0.5561. From the distribution of their data, this may be because the data of task 1c is not balanced. Moreover, because task 1c has a dependency on task 1a, only filling in missing values with 0 may affect the judgment of the system.

Model	F1-Score
RoBERTa	0.80
RoBERTa+BiGRU	0.86
RoBERTa+BiGRU+BiLSTM	0.88

Table 4: The comparative results of task 1c, and the model is the *base* version.

task	label 0	label 1
subtask 1a	386	614
subtask 1c	331	669

Table 5: The result distribution of task 1b and 2

### 4.2 Regression results

The output of the regression result is shown in Figure 4. In subtask 1b, all predicted values are between 0 and 3. Most of the values are in the range

of 2 to 3 (412 sentences). In subtask 2, all predicted values are also between 0 and 3 (442 sentences score 0). Comparing with the data distribution in the training set, we find that the distribution of the predicted value is consistent with the distribution of the training data.

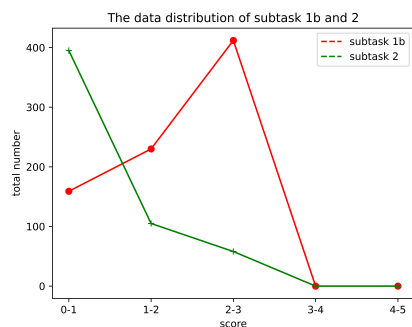


Figure 4: The predicting result of *humor\_rating*

## 5 Conclusion and Future Work

In this work, I present my result on HaHackathon: Detecting and Rating Humor and Offense which includes four subtasks. For tasks 1a and 1c, I use the BiGRU-Attention based on the ALBERT model to complete subtask 1a, and 1c is completed by RoBERTa combine with BiLSTM+BiGRU and this model works well. I also summarized the possible reasons for the low score in task 1c.

From a theoretical and computational point of view, it is difficult to establish a mechanism for computers to understand humor like humans. The reason is as follows. 1) The definition of humor is loose. It is almost impossible to identify humor by establishing rules. 2) Humor is related to context and background. Humor expects to break the common sense of readers in a specific situation. In the future, we should design features that are interpretable, calculable, and easy to implement that conform to humor theory.

## References

Zsolt Bitvai and Trevor Cohn. 2013. Non-linear text regression with a deep convolutional neural network. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics:180–185.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805.

Yadan Fan, Sicheng Zhou, Yifan Li, and Rui Zhang. 2019. Deep learning approaches for extracting adverse events and indications of dietary supplements from clinical text. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers).

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, and Piyush Sharma. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. arXiv:1909.11942. Version 6.

Lizhen Liu, Donghai Zhang, and Wei Song. 2018. Exploiting syntactic structures for humor recognition. Proceedings of the 27th International Conference on Computational Linguistics:1875–1883.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, and Mandar Joshi. 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692. Version 1.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Alex Morales and Chengxiang Zhai. 2017. Identifying humor in reviews using background text sources. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.

Shivashankar Subramanian, Timothy Baldwin, and Trevor Cohn. 2018. Content-based popularity prediction of online petitions using a deep regression model. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers):182–188.

Qiang wei, Zongcheng Ji, and zhiheng Li. 2020. A study of deep learning approaches for medication and adverse drug event extraction from clinical text. *Journal of the American Medical Informatics Association*, 27(1), 2020, 13–21.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. Humor recognition and humor anchor extraction. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.

M. Zampieri, S. Malmasi, P. Nakov, and S. Rosenthal. 2019. Predicting the type and target of offensive posts in social media. arXiv:1902.09666.



# DUTH at SemEval-2021 Task 7: Is Conventional Machine Learning for Humorous and Offensive Tasks enough in 2021?

Alexandros Karasakalidis    Dimitrios Effrosynidis    Avi Arampatzis

Database & Information Retrieval research unit,  
Department of Electrical & Computer Engineering,  
Democritus University of Thrace, Xanthi 67100, Greece  
{alexkara23, deffrosy, avi}@ee.duth.gr

## Abstract

This paper describes the approach that was developed for SemEval 2021 Task 7 Hackathon: Incorporating Demographic Factors into Shared Humor Tasks (Meaney et al., 2021) by the DUTH Team. We used and compared a variety of preprocessing techniques, vectorization methods, and numerous conventional machine learning algorithms, in order to construct classification and regression models for the given tasks. We used majority voting to combine the models' outputs with small Neural Networks (NN) for classification tasks and their mean for regression for improving our system's performance. While these methods proved weaker than modern, deep learning models, they are still relevant in research tasks because of their low requirements on computational power and faster training.

## 1 Introduction

The underpinnings of humor have proven far more vexing than those of other emotional experiences. It is a highly subjective topic that various scholars have attempted to construct theories for understanding its fundamental elements in the studies of philosophy, linguistics, psychology and sociology. Some theories, e.g. the *Benign Violation Theory* (Warren and McGraw, 2015), suggest that humor can be described as linguistic violations that still make grammatical sense. The aforementioned theory supports that for a joke to be classified as humorous, it needs to avoid being too harmless or too offensive.

There are numerous studies in humor sentiment analysis in the last decade. In microblogging, Reyes et al. (2012) considered extracting linguistic devices from tweets to be used as features for classifying these tweets as humorous or ironic, while Raz (2012) approached the classification of humorous tweets as a multi-class problem of 11 types of

humor, in his attempts to better attribute the real sentiment of a tweet. Recent attempts on humor detection on SemEval's 2020 Task 7 indicate that transformer models like BERT (Mahurkar and Patil, 2020) far outperform traditional machine learning algorithms (S et al., 2020).

This paper describes our submissions to SemEval 2021 task 7 and is structured as follows. Section 2 describes the tasks, training data, and evaluation measures. Section 3 describes key methods and algorithms used. Section 4 describes our proposed system while Section 5 analyzes our results. Finally, we draw our conclusions in Section 6, where we also propose directions for future work.

## 2 Background

In this section we describe each subtask's objective, the given data, and evaluation measures.

### 2.1 Subtasks

The main objectives of SemEval 2021 Task 7 were split into 4 subtasks. Subtask 1a required us to classify short texts as humorous or not, while Subtasks 1b and 1c required us to rate the text's humor and further classify it as controversial or not respectively. Finally, in Subtask 2, we had to rate how offensive each text was—humorous or not. All texts were in English.

### 2.2 Dataset

The organizers released the full training data in three parts: trial, development, and evaluation. Our final training dataset consisted of 9,000 different texts annotated with labels regarding each subtask in csv file format, while our test set consisted of 1,000 texts. Statistics for the training dataset are presented in Table 1 and Figure 1.

	Is humorous?	Is controversial?
Yes	5,564	2,773
No	3,436	2,791

Table 1: Humor and controversiality labels in the training set

### 2.3 Evaluation Measures

For the classification Subtasks 1a and 1c, we use the  $F_1$  measure. For the regression Subtasks 1b and 2, we use the root mean squared error (RMSE).

## 3 Experimental Setup

In this section we describe the preprocessing and vectorization methods as well as the machine learning algorithms used.

### 3.1 Preprocessing

Text preprocessing is the backbone of every text classification task. We applied the following techniques:

1. Tokenizing and Lowercasing words: We lower-cased the words in the texts and split them into tokens.
2. Stemming or Lemmatisation: Reducing noise from texts while (generally) improving system performance.
3. Removing Stopwords: Stopwords do not add much meaning to a sentence, so removing them helps in reducing the number of features and improving results.
4. Tagging words with capital letters: Tagging each word containing a capital letter that is not the first word in a sentence. Applied (when used) prior to word lowercasing.
5. Replacing Emojis: Very few emojis were found in texts, so we replaced them with their corresponding sentiment.
6. Replacing Contractions: We replace contractions into their full forms using a dictionary.
7. Removing integers: Numbers have no emotional value, so we remove them.
8. Part-of-Speech (POS) tagging: We used POS tagging of words for preprocessing the texts, following two different approaches.

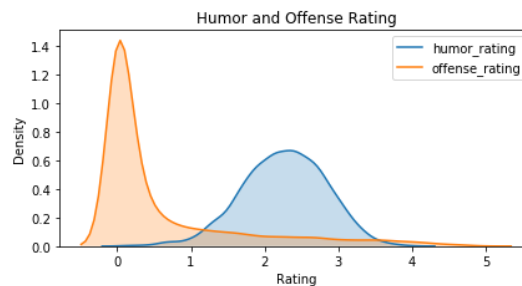


Figure 1: Humor and Offense Rating density plot

- (a) Appending POS tags in words: Aims to incorporate part-of-speech information in our features.
- (b) Removing words based on their POS tag: Aims to remove words with low sentimental value from the data by targeting specific tags.

9. Numeric Feature Extraction: We extracted counts of characters, words, exclamation points, and numbers, as well as the numbers of declarative, interrogative, and imperative/exclamative sentences in a text. Finally, we extracted the counts of verbs, nouns, and adjectives, for each text.

The performance comparison of each preprocessing method is shown in Section 5. Stemming, lemmatisation, POS tagging, and most of numeric feature extractions were achieved by using tools from the well established NLTK (Elhadad, 2010), while we were guided by the survey of Ravi and Ravi (2015) of the most commonly used techniques in text preprocessing for sentiment analysis and by our previous works (Effrosynidis et al., 2017; Symeonidis et al., 2018) on this subject.

### 3.2 Machine Learning

The training of our classification and regression models aimed to improve the evaluation measure used for the corresponding task. Many probabilistic, linear and tree-based algorithms were used, as well as small neural network architectures.

The algorithms/Neural Networks (NN) that performed the best were used in our systems and are listed below:

- Linear Models: Linear SVM, Bayesian Ridge Regression and LASSO
- Non-Linear Models: Naive Bayes, Light GBM (Ke et al., 2017) and XGBoost (Chen and Guestrin, 2016)

- NN Models: Dense and Long Short-Term Memory Networks (using the Keras API<sup>1</sup>)

Linear SVM models were excluded from our final systems mainly due to our better tuning of the LGBM and XGB models during the last phase.

### 3.3 Vectorization and Embedding

We used the following scikit-learn toolkit’s vectorizers to extract features from the preprocessed data using word unigrams or bigrams:

- Tf-idf Vectorizer: translates the word counts matrix to a matrix of tf-idf features.
- Delta tf-idf Vectorizer: proposed by [Martineau and Finin \(2009\)](#), it creates tf-idf features similarly to tf-idf vectorizer but applies a weighting scheme reflecting the difference of tf-idf value of each word between the texts of two classes. We used the subtask’s 1a labels for weighting tf-idf values in every subtask.

In order to create features for the LSTM models, we translate the words of each text into word vectors. This translation is achieved through the use of a well-known, pre-trained model: GloVe ([Pennington et al., 2014](#)).

## 4 System Overview

In this section we describe the proposed system for each subtask.

### 4.1 Proposed System

We trained each model with all possible combinations of preprocessing and vectorization. During the development phase, we evaluated these models using 10-fold cross validation on the training data. These evaluations guided us through hyperparameter tuning and model selection.

During the evaluation phase, we combined the outputs of these models in order to produce our system’s predictions (Figure 2). The system’s performance was evaluated using the test data from the development phase. That data was also used as validation data for training and tuning the dense and LSTM networks.

Model selection for our final systems was a repetitive but simple process. We selected the best performing model per algorithm and vectorization method, some weaker models whose outputs had

<sup>1</sup><https://github.com/fchollet/keras>

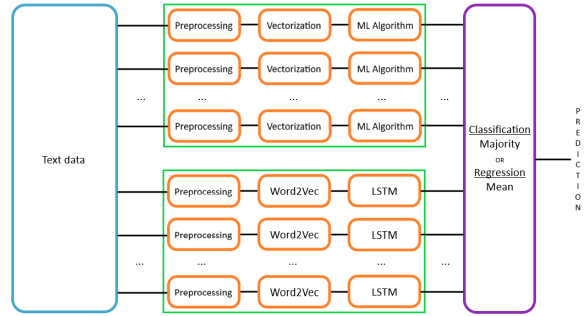


Figure 2: System Architecture

lower correlation than the outputs of the best performing models and all the NN models that we had trained. We then combined ([Symeonidis et al., 2017](#)) all selected model outputs in all possible combinations consisting of at least 3 models and picked the best performing one as our final system for task prediction.

This process was repeated for each subtask. After finding the combination that produced the best results for the development’s phase test data, we re-trained the system’s models by appending that test data on the training data.

#### 4.1.1 Subtask 1a: Humour Classification

In Subtask 1a, each text needs to be classified as humorous or not. Table 2 showcases the various preprocessing methods, vectorization tools and ML algorithms that comprised our final system. Since the number of models is even, majority voting favors the ‘non-humorous’ label in case of a tie, i.e. the less represented tag in the dataset.

Preprocessing	Vectorizer	ML
1,2,5,8a and 9	Delta tf-idf unigrams	Shallow NN
1,2 and 9	Tf-idf unigrams	LGBM
1,2,8a and 9	Delta tf-idf bigrams	Naive Bayes
1,2,6,7,8a and 9	Delta tf-idf bigrams	Naive Bayes
1,7 and 9	Tf-idf bigrams	XGB
1	Word2vec embeddings	LSTM

Table 2: Subtask 1a system composition

#### 4.1.2 Subtask 1b: Humor Rating

In Subtask 1b, each humorous text needs to be rated in a range of 0–5 on how much humorous it is. For this subtask, the best combination found amounted to 13 models. Thus, we will not include a table for this task. All proposed preprocessing techniques were used but 3 and 8b as well as every vectorizer. Interestingly, NN models were ruled out in this

subtask since a LGBM, XGB and Bayesian Ridge combination produced the best outcome.

### 4.1.3 Subtask 1c: Controversial Humor Classification

In Subtask 1c, each humorous text has to be classified as controversial or not. This is the only task that a single model outperformed any combination of models we tried to assemble. A preprocess of extracting numeric features (9), appending POS tags (8a), lowercasing and tokenization (1), a delta tf-idf vectorizer extracting bigrams and an LGBM model outperformed every other combination.

### 4.1.4 Subtask 2: Offense Rating

Finally, in Subtask 2 each text is rated in the range of 0–5 on how offensive it is. The final system is the average of the 7 individual models described in Table 3.

Preprocessing	Vectorizer	ML
1,2 and 9	delta tf-idf unigrams	XGB
1,2,3 and 9	tf-idf biwords	XGB
1	word2vec embeddings	LSTM
1 and 2	word2vec embeddings	LSTM
1 and 3	word2vec embeddings	LSTM
1 and 5	word2vec embeddings	LSTM
1 and 7	word2vec embeddings	LSTM

Table 3: Subask 2 system composition

## 5 Results

Each preprocessing method had an impact on model performance, as it is shown in Table 4 for each task.

The results for each task/subtask are shown in Table 5. We present the scores of our best performing single models and combination of models respectively on the development test set as well as our submissions for the evaluation test data.

We can detect a pattern in the difference between the winning team’s submissions and our submissions, and between the performance of our NN and non-NN models. It would be an indication—for Subtask 1a—that our conventional machine learning models, while achieving a respectable performance, cannot handle some outliers. This can be also observed through the results of Subtask 2, where outliers have a greater impact on the RMSE metric. Our false negative results on humor are mostly ironic, reference-based jokes or highly controversial ones, and our false positive

Preprocessing	Subtask 1a	Subtask 1b	Subtask 1c	Subtask 2
None	0.8325	0.5501	0.6299	0.8431
Stemming	<b>0.8372</b>	<b>0.5500</b>	0.6266	0.8560
Lemmatization	<b>0.8361</b>	0.5535	0.6130	0.8504
Stopwords	0.8206	<b>0.5465</b>	<b>0.6379</b>	0.8623
Capital Tagging	0.8284	<b>0.5550</b>	0.6099	0.8790
Emoji Replace	<b>0.8332</b>	0.5501	0.6299	0.8492
Contraction Replace	<b>0.8345</b>	<b>0.5471</b>	0.6253	0.8587
Integers Remove	<b>0.8357</b>	0.5567	0.6176	0.8664
POS Tag Append	<b>0.8331</b>	<b>0.5459</b>	<b>0.6455</b>	0.8620
POS Tag Filter	0.8170	0.5558	0.6214	1.0022
Feature Creation	<b>0.8457</b>	<b>0.5473</b>	0.5298	0.8653

Table 4:  $F_1$  or RMSE of best performing model per preprocessing method. ‘None’ stands for plain tokenization and lowercasing. Scores in **bold** are better than ‘None’.

results are mostly conversational writing texts like microblogging posts. The basic LSTM models we created were able to slightly close the gap with the superior, transformer models but there is still plenty of headroom for improvement.

On the other hand, our systems on Subtasks 1b and 1c were much closer to their superior models. While Subtask’s 1b results could be attributed to a large extend on the distribution of humor ratings, Subtask 1c seems to be a much harder task regardless the approach. With an average accuracy of 0.5 across all submissions, humor controversiality seems to puzzle even the most complex models; anyway, humor controversiality is much more subjective than humor itself.

Nevertheless, a great advantage of conventional machine learning is training speed and hardware requirements. State-of-the-art boosting models like the ones we used (LightGBM and XGB) can be accelerated through the use of GPUs while our small NNs can be trained in a couple of minutes. Training/tuning deep learning models, on the other hand, requires expensive hardware and can be very time-consuming.

## 6 Conclusions

In this report, we presented our approach on humorous and offensive text classification and rating based on the combination of outputs from different preprocessing techniques, vectorization methods and machine learning algorithms. Our proposed systems were outperformed by other teams in the main tasks, while our conventional machine learning models were mostly inferior to our neural networks. Our future work will focus on expanding our preprocessing methods, introducing further ensemble methods and stacking, as well as

	Development Test			Evaluation Test		
	Best Single non-NN Model	Best Single NN Model	Best System	Best Submission	Our Position	Winning Submission
Subtask 1a	0.8706 ( $F_1$ )	0.8952 ( $F_1$ )	0.9136 ( $F_1$ )	0.8942 ( $F_1$ )	52 out of 58	0.9820 ( $F_1$ )
Subtask 1b	0.5411 (RMSE)	0.5492 (RMSE)	0.5411 (RMSE)	0.5507 (RMSE)	12 out of 50	0.4959 (RMSE)
Subtask 1c	0.6455 ( $F_1$ )	0.6491 ( $F_1$ )	0.6636 ( $F_1$ )	0.5990 ( $F_1$ )	12 out of 36	0.6302 ( $F_1$ )
Subtask 2	0.8313 (RMSE)	0.7552 (RMSE)	0.7368 (RMSE)	0.5819 (RMSE)	40 out of 48	0.4120 (RMSE)

Table 5: Model/System performance in development and evaluation test sets

transformer-based models for direct comparisons.

## References

- Tianqi Chen and Carlos Guestrin. 2016. **Xgboost: A scalable tree boosting system**. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 785–794. ACM.
- Dimitrios Effrosynidis, Symeon Symeonidis, and Avi Arampatzis. 2017. **A comparison of pre-processing techniques for twitter sentiment analysis**. In *Research and Advanced Technology for Digital Libraries - 21st International Conference on Theory and Practice of Digital Libraries, TPDL 2017, Thessaloniki, Greece, September 18-21, 2017, Proceedings*, volume 10450 of *Lecture Notes in Computer Science*, pages 394–406. Springer.
- Michael Elhadad. 2010. **Natural language processing with python** steven bird, ewan klein, and edward looper (university of melbourne, university of edinburgh, and BBN technologies) sebastopol, CA: o’reilly media, 2009, xx+482 pp; paperbound, ISBN 978-0-596-51649-9, \$44.99; on-line free of charge at [nltk.org/book](http://nltk.org/book). *Comput. Linguistics*, 36(4):767–771.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. **Lightgbm: A highly efficient gradient boosting decision tree**. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3146–3154.
- Siddhant Mahurkar and Rajaswa Patil. 2020. **LRG at semeval-2020 task 7: Assessing the ability of BERT and derivative models to perform short-edits based humor grading**. *CoRR*, abs/2006.00607.
- Justin Martineau and Tim Finin. 2009. **Delta TFIDF: an improved feature space for sentiment analysis**. In *Proceedings of the Third International Conference on Weblogs and Social Media, ICWSM 2009, San Jose, California, USA, May 17-20, 2009*. The AAAI Press.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. **Semeval 2021 task 7, hahackathon, detecting and rating humor and offense**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. **Glove: Global vectors for word representation**. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL.
- Kumar Ravi and Vadlamani Ravi. 2015. **A survey on opinion mining and sentiment analysis: Tasks, approaches and applications**. *Knowl. Based Syst.*, 89:14–46.
- Yishay Raz. 2012. **Automatic humor classification on twitter**. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 3-8, 2012, Montréal, Canada*, pages 66–70. The Association for Computational Linguistics.
- Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. **From humor recognition to irony detection: The figurative language of social media**. *Data Knowl. Eng.*, 74:1–12.
- Kayalvizhi S, Thenmozhi D., and Aravindan Chandrabose. 2020. **SSN\_NLP at SemEval-2020 task 7: Detecting funniness level using traditional learning with sentence embeddings**. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 865–870, Barcelona (online). International Committee for Computational Linguistics.
- Symeon Symeonidis, Dimitrios Effrosynidis, and Avi Arampatzis. 2018. **A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis**. *Expert Syst. Appl.*, 110:298–310.
- Symeon Symeonidis, Dimitrios Effrosynidis, John Kordonis, and Avi Arampatzis. 2017. **DUTH at semeval-2017 task 4: A voting classification approach for twitter sentiment analysis**. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 704–708. Association for Computational Linguistics.
- Caleb Warren and A. Peter McGraw. 2015. **Benign violation theory**. *Mays Business School*, 2015-11.

# DeepBlueAI at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with Stacking Diverse Language Model-Based Methods

Bingyan Song Chunguang Pan Shengguang Wang Zhipeng Luo  
DeepBlue Technology (Shanghai) Co., Ltd  
{songby, panchg, wangshg, luozp}@deepblueai.com

## Abstract

This paper describes the winning system for SemEval-2021 Task 7: Detecting and Rating Humor and Offense. Our strategy is stacking diverse pre-trained language models (PLMs) such as RoBERTa and ALBERT. We first perform fine-tuning on these two PLMs with various hyperparameters and different training strategies. Then a valid stacking mechanism is applied on top of the fine-tuned PLMs to get the final prediction. Experimental results on the dataset released by the organizer of the task show the validity of our method and we win first place and third place for subtask 2 and 1a.

## 1 Introduction

Humor and offense detection continue to be challenging AI problems since humor and offense involve in-depth world-knowledge, common sense, and the ability to perceive relationships across entities and objects at various levels of understanding (Hossain et al., 2019). The recognition of humor and offense in the text has been receiving much attention (Zampieri et al., 2019; Hossain et al., 2020). Accordingly, SemEval-2021 Task 7, **Detecting and Rating Humor and Offense**, which aims to automatically recognize humor in English jokes was held (Meaney et al., 2021).

In this paper, we introduce our system for accomplishing the above task by leveraging pre-trained models (PLMs). There are two main steps for our system, i) fine-tuning two kinds of PLMs, including ALBERT (Lan et al., 2019) and RoBERTa (Liu et al., 2019) with various hyperparameters and training strategies, achieving diverse models; ii) applying a validity stacking mechanism on top of these PLMs to do the final predictions.

Our experimental results show that merging PLMs with different training strategies together can achieve great improvement which verifies the effectiveness of increasing model diversity. As a

Tags	No. of is_humor	Percentage
train	4932	61.65%
dev	632	63.20%
test	615	61.50%

Table 1: The number and percentage of humor samples in training set, validation set and test set respectively.

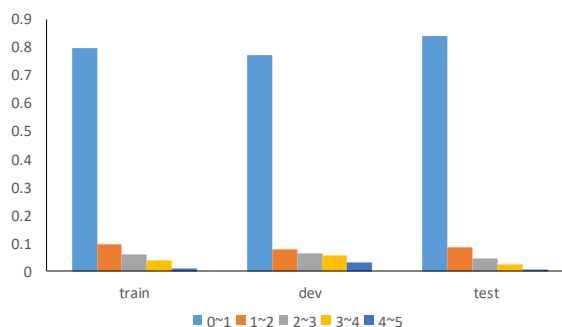


Figure 1: The distributions of offense rating for training set, validation set and test set.

result, our system achieves the F1-score of 96.76% in subtask 1a and the RMSE of 41.2% in subtask 2, which ranks third and first among all the participated teams respectively.

## 2 Background

### 2.1 Task Definition

The “Detecting and Rating Humor and Offense” task, shared by SemEval-2021, consists of two subtasks. Subtask 1 includes three parts, a) A binary task to predict if the text would be considered humorous; b) A regression task to predict how humorous a text is if it is classed as humorous; c) A binary task to predict if the humor rating would be considered controversial when the text is classed as humorous. Subtask 2 aims to predict how offensive a text would be with values between 0 and 5. This score can be calculated regardless of whether the text is classed as humorous or not. In this paper, we mainly focus on subtask 1a and subtask 2.

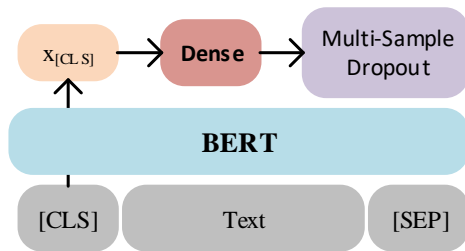


Figure 2: The overall architecture for detecting and rating humor and offense.

## 2.2 Dataset

Humor and Offense appreciation is a highly subjective phenomenon, with age, gender, race, and socioeconomic status are known to have an impact on the perception of a joke. The labels and ratings of the English dataset in this task are collected from a balanced set of age groups from 18-70 and are various in genders, political stances, and income levels. The dataset has a total of 10,000 samples, which are divided into training set, validation set, and test set according to 8:1:1. Table 1 shows the number and percentage of humor samples in the three datasets and we can find that their distributions are very similar with nearly 60% are humor ones. Figure 1 demonstrates the distribution of offense ratings in three datasets. Samples with offense ratings between (0,1) are the most and the three datasets have the same distribution of offense ratings as well.

## 3 System Overview

### 3.1 PLMs-based Method

**Architecture** In our method, we have the same architecture for dealing with subtask 1a and subtask 2. As shown in Figure 2, we utilize several pre-trained language models (e.g., RoBERTa) as the encoder and segment different texts with special tokens [CLS] and [SEP]. After the tokenization, we can get the embedding of [CLS], which can be seen as the representation for the whole input text. We pass it through a dense layer and obtain the final prediction through the Multi-Sample Dropout (Inoue, 2019). The output of dense layer  $x$  is depicted as below,

$$x = ReLU(W_0 dropout(x_{[CLS]})) \quad (1)$$

where  $W_0 \in R^{d \times k}$  is the learning weight,  $k$  is the dimension of  $e_{[CLS]}$  and  $d$  is a hyperparameter

which we set as 256 and the dropout rate here we set as 0.2 or 0.5.

**Multi-Sample Dropout** Dropout is a simple but efficient regularization technique for achieving better generalization of deep neural networks. During training, dropout randomly discards a portion of the neurons to avoid overfitting. The original dropout creates a randomly selected subset (called a dropout sample) from the input in each training iteration while the multi-sample dropout creates multiple dropout samples. The loss is calculated for each sample, and the sample losses are averaged to obtain the final loss.

Thus, the final prediction of both subtask 1a and 2 can be computed as follows,

$$\hat{y} = \frac{1}{N} \sum_{i=1}^N Sigmoid(W_i dropout_i(x)) \quad (2)$$

where  $W_i \in R^{1 \times d}$  is the learning weights,  $N$  is the number of dropout values which we set as 5. By using this training mechanism, we can accelerate training and achieve lower error rates as well. Since the Sigmoid function used here is the logistic function which maps any real value to the range (0,1), we preprocess the rating of offense in subtask 2 from (0,5) to (0,1).

**Loss function** As mentioned above, subtask 1a is a binary task and subtask 2 is a regression task, thus we choose Binary Cross Entropy (BCE) and Mean Square Error (MSE) as the loss function respectively.

### 3.2 Training strategies

To further improve the diversity and accuracy of trained models, we incorporate three training strategies as depicted below.

**Task-Adaptive Pre-training** Task-adaptive pre-training (TAPT) is an effective method to improve model performance (Gururangan et al., 2020). The data used in general pre-training usually vary from task-specific data. Thus we do task-adaptive by pre-training the masked language model task on the given Humor and Offense dataset.

**Pseudo-Labeling** Pseudo labeling (PL) is the process of using a labeled data model to predict labels for unlabeled data. We predict the unlabeled test dataset and mix these pseudo labels with the training set together to train the new model. For

Subtask 1a		Subtask 2	
Model	F1	Model	RMSE
ALBERT <sub>BASE</sub>	0.9635	-	-
ALBERT <sub>BASE+AT</sub>	0.9662	-	-
RoBERTa <sub>LARGE</sub>	0.9685	RoBERTa <sub>LARGE</sub>	0.4846
RoBERTa <sub>LARGE+AT</sub>	0.9694	RoBERTa <sub>LARGE+AT</sub>	0.4713
RoBERTa <sub>LARGE+TAPT</sub>	0.9724	RoBERTa <sub>LARGE+TPAT</sub>	0.4621
RoBERTa <sub>LARGE+TAPT+AT</sub>	0.9727	RoBERTa <sub>LARGE+TAPT+AT</sub>	0.4607
RoBERTa <sub>LARGE+TAPT+KD</sub>	0.9714	RoBERTa <sub>LARGE+TAPT+KD</sub>	0.4633
RoBERTa <sub>LARGE+TAPT+KD+AT</sub>	0.9726	RoBERTa <sub>LARGE+TAPT+KD+AT</sub>	0.4605
RoBERTa <sub>LARGE+TAPT+PL</sub>	0.9728	RoBERTa <sub>LARGE+TAPT+PL</sub>	0.4571
<b>RoBERTa<sub>LARGE+TAPT+PL+AT</sub></b>	<b>0.9738</b>	<b>RoBERTa<sub>LARGE+TAPT+PL+AT</sub></b>	<b>0.456</b>

Table 2: Comparison of pre-trained language models with different training strategies of Subtask 1a and 2.

subtask 1a, we set the threshold as 0.8 which means samples with predicted scores higher than 0.8 are treated as the humor ones.

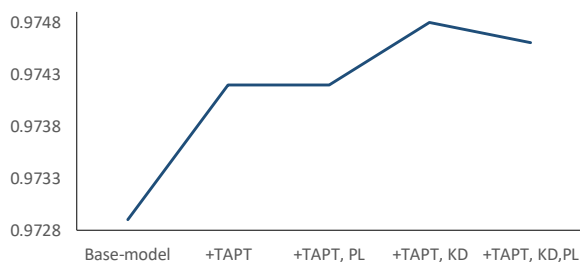


Figure 3: The comparison of F1 scores for stacking different models in subtask 1a.

**Knowledge Distillation** Inspired by (Hinton et al., 2015), we adopt the knowledge distillation (KD) mechanism into our system. The whole procedure consists of three steps. First, we train the original big model using a hard target, which is the true label given in the dataset. Next, we use the trained model to predict the soft target, which is the probability for each sample being humorous and offense. After this, we train a small model by minimizing the loss between the scores predicted by the small model and the soft target. The loss functions are still BCE and MSE. At last, we use the small model to predict the final results.

**Adversarial Training** Adversarial training (AT) is a popular approach to increasing the robustness of neural networks and has good regularization performance (Miyato et al., 2016). By adding perturbations to the embedding layer, we can get more stable word representations and a more generalized model, which significantly improves model performance on unseen data.

### 3.3 Stacking Trained Models

Model stacking is an efficient ensemble method to improve model accuracy. The main procedure

of stacking trained models in our method including five steps. First, we use two different PLMs including RoBERTa and ALBERT. Second, we do TAPT on these PLMs to achieve new pre-trained models. Third, we perform 7-fold cross-validation on the whole training process to avoid overfitting or selection bias. Fourth, we train various models with different hyperparameters and different training strategies to improve the model diversity. Ultimately, we average all the predictions from different models to get the final prediction.

## 4 Experiments

**Evaluation Metrics** As mentioned in the official evaluation procedure of SemEval-2021 task 7, the main evaluation metrics for the binary classification tasks is f1-measure and the metric for the regression tasks is Root Mean Squared Error (RMSE).

**Parameter settings** All models are implemented based on the open-source transformers library of hugging face (Wolf et al., 2020), which provides thousands of pre-trained models that can be quickly downloaded and fine-tuned on specific tasks. To do better performance estimation, We gather the training set and validation set together as the new training set and then do 7-fold cross-validation on it. We set batch size as 16 and run 10 epochs for each fold. The learning rate is 1e-5. For RoBERTa<sub>LARGE</sub> and ALBERT<sub>BASE</sub>, the k is set as 1024 and 128 respectively.

## 5 Results

### 5.1 Ablation Studies

**PLMs with Training Strategies** For subtask 1a, we use two types of PLMs including ALBERT<sub>BASE</sub> and RoBERTa<sub>LARGE</sub>. As shown in Table 2. We set five groups of models and each group is the same models with or without adversarial training (AD).



The models of the first and second groups are the base ones and we add training strategies including task-adaptive pre-training (TAPT), knowledge distillation (KD), and pseudo-labeling (PL) to the other three groups.

The results are the average scores from models with different hyperparameters (e.g. different dropout) by doing 7-fold cross-validation on the new training dataset depicted above. Since RoBERTa<sub>LARGE</sub> performs better on this task, ALBERT is not used in subtask 2 anymore. From Table 2, we find that for both subtask 1a and subtask 2, all the training strategies can improve the performance. Besides, models with AD achieve better scores than the ones without AD. The models adding TAPT, PL and AT together are the best ones.

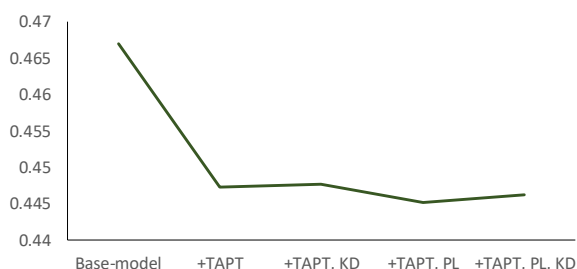


Figure 4: The comparison of RMSE for stacking different models in subtask 2.

**Stacking trained models** To stack the trained models, we use a simple method which averaging predictions from different models. Figure 3 and 4 show the comparison for stacking different models of subtask 1a and 2. We find that all scores of the ensemble ones are better than the best score in Table 2 which from a single model. This verifies the effectiveness of stacking different models.

However, both Figure 3 and 4 demonstrate that the best score is not to stacking models of all the groups in Table 2 but to stack part of the models. This indicates that combining the least correlated results is more efficient than combining them all.

Subtask 1a		Subtask 2	
System	F1	System	RMSE
endworld	0.9854	<b>DeepBlueAI</b>	<b>0.412</b>
stee	0.9797	m m m m	0.419
<b>DeepBlueAI</b>	<b>0.9676</b>	calamity link	0.423
baseline	0.9283	baseline	0.5770

Table 3: Leaderboard

## 5.2 Official Ranking

We submitted the scores predicted by the ensemble method introduced above. The official ranking is presented in Table 3. We rank third in subtask 1a and first in subtask 2, which verifies the validity of our system.

## 6 Conclusion

In this paper, we propose a top-performing approach for the task of Detecting and Rating Humor and Offense. We fine-tune two kinds of pre-trained language models including ALBERT and RoBERTa with different training strategies such as pseudo labeling and knowledge distillation. Then, we stack them with a simple linear regression model. Experimental results show the effectiveness of this ensemble method and we win first place and third place for subtask 2 and 1a. For future work, it would be interesting to test the performance of our best-performing system on other humor detection datasets to validate its portability and robustness.

## References

- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Nabil Hossain, John Krumm, and Michael Gamon. 2019. “president vows to cut; taxes; hair”: Dataset and analysis of creative text editing for humorous headlines. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 133–142.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. Semeval-2020 task 7: Assessing humor in edited news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758.
- Hiroshi Inoue. 2019. Multi-sample dropout for accelerated training and better generalization. *arXiv preprint arXiv:1905.09788*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised

- learning of language representations. In *International Conference on Learning Representations*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*.
- Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

# CS-UM6P at SemEval-2021 Task 7: Deep Multi-Task Learning Model for Detecting and Rating Humor and Offense

Kabil Essefar<sup>1</sup>   Abdellah El Mekki<sup>1</sup>   Abdelkader El Mahdaouy<sup>1</sup>  
Nabil El Mamoun<sup>2</sup>   Ismail Berrada<sup>1</sup>

<sup>1</sup>School of Computer Sciences, Mohammed VI Polytechnic University, Morocco

<sup>2</sup>Faculty of Sciences Dhar EL Mahraz, Sidi Mohamed Ben Abdellah University, Morocco  
{firstname.lastname}@um6p.ma

## Abstract

Humor detection has become a topic of interest for several research teams, especially those involved in socio-psychological studies, with the aim to detect the humor and the temper of a targeted population (e.g. a community, a city, a country, the employees of a given company). Most of the existing studies have formulated the humor detection problem as a binary classification task, whereas it revolves around learning the sense of humor by evaluating its different degrees. In this paper, we propose an end-to-end deep Multi-Task Learning (MTL) model to detect and rate humor and offense. It consists of a pre-trained transformer encoder and task-specific attention layers. The model is trained using MTL uncertainty loss weighting to adaptively combine all sub-tasks objective functions. Our MTL model tackles all sub-tasks of the SemEval-2021 Task-7 in one end-to-end deep learning system and shows very promising results.

## 1 Introduction

Humor is a human trait that defines the emotional and behavioral characteristics of an individual. It refers to the quality of being amusing, comic, sarcastic, etc. Most dictionaries define humor also as a message, whose ingenuity or verbal skill, or incongruity, that has the power to make individual laughing.

Humor and offensive language detection tasks are increasingly becoming hot research topics in Natural Language Processing (NLP) (Zampieri et al., 2019; Reyes et al., 2012; Gleason et al., 2019; van den Beukel and Aroyo, 2018; Cattle and Ma, 2018; Singh et al., 2020). Existing research works have tackled humor detection as either a binary classification problem (Weller and Seppi, 2019; Annamoradnejad, 2020) or a ranking task (Potash et al., 2017; Hossain et al., 2020; Zhang

et al., 2019). Similarly, most research works on offensive language detection have proposed methods and approaches to discriminate between offensive and not-offensive texts (Zampieri et al., 2019, 2020), whereas, other research works have classified offensive content into more fine-grained levels (Wiegand et al., 2018; Kumar et al., 2018; Risch et al., 2020).

Fine-tuning pre-trained transformer-based language models on the target task data has shown state-of-the-art (SOTA) results in many NLP applications (Devlin et al., 2019; Liu et al., 2019). For instance, several research works on humor and offensive language detection have achieved SOTA performances using pre-trained transformer-based language models (Zampieri et al., 2019; Weller and Seppi, 2019; Zampieri et al., 2020).

In this paper, we describe our system submitted to the SemEval-2021 Task-7 (Sub-Tasks 1 and 2) (Meaney et al., 2021). We propose an end-to-end deep Multi-Task Learning (MTL) model based on RoBERTa Encoder (Liu et al., 2019) and task-specific attention layers. The attention mechanism is applied on top of the encoder’s contextualized word embedding to extract task-specific features. The classification and regression modules are fed with their task-specific attention output and the shared pooled output of the encoder. In order to adaptively combine all tasks’ losses, we employed the MTL uncertainty loss weighting method (Kendall et al., 2017). We also investigate the base and the large variants of BERT (Devlin et al., 2019) and RoBERTa encoders for both single-task and MTL. The obtained results show that our MTL model outperforms its single-task counterparts on both Task 1 and Task2. The best performances are obtained using RoBERTa-large encoder. Our system is ranked 18th, 9th, 7th and 20th on Sub-Tasks 1a, 1b, 1c and 2a, respectively.

The remainder of this paper is organized as fol-

lows. Section 2 describes the SemEval-2021 task-7 and the provided data. Section 3 presents our MTL system. Section 4 summarizes the obtained results. Section 5 concludes the paper.

## 2 Task description

The SemEval-2021 Task 7 consists of two main tasks: the first task seeks recognizing and rating humor while the second task aims to rate offense (Meaney et al., 2021). To this end, the organizers have provided 8000 sentences for the training, and 1000 sentences for the validation and test. All training and validation sentences are labeled for humor detection and offense rating, while only humorous sentences are labeled for humor and controversy rating. The dataset is labeled by 20 annotators. They have a balanced set of age groups from 18 to 70.

### 2.1 Task 1: Humor detection

The aim is to predict four target values for the following sub-tasks:

- Task 1a: This sub-task is a binary classification task where the aim is to classify texts as humorous or not.
- Task 1b: This sub-task consists of predicting the humor degree of a text. The degree is based on the average rating (from 0 to 5) given by the annotators.
- Task 1c: This sub-task consists of predicting whether the humor rating would be considered controversial or not: i.e. whether or not the variance between the annotators’ ratings is higher than the median rating.

### 2.2 Task 2: Offensive rating

This task has one sub-task for offense rating:

- Task 2a: This task predicts the degree of offense conveyed in a text regardless of its humor label. The offense degree varies from 0 (not offensive) to 5 (very offensive).

## 3 System description

We propose an end-to-end deep MTL model based on pre-trained transformer-based language model (Devlin et al., 2019; Liu et al., 2019) and task-specific attention layers. First, we apply the encoder to the input text in order to obtain its Contextual Word Embedding (CWE). The task-specific

attention layers are applied on the CWE. The classifier (Task 1a, Task 1c) or the regressor (Task 1b, Task 2a) is fed with the concatenation of its task-specific attention output and the encoder’s pooled output. The model is then trained to minimize the binary cross-entropy loss and the RMSE loss for the classification and regression tasks, respectively. Finally, these losses are combined using uncertainty loss weighting for MTL.

### 3.1 Transformer encoder

In order to recognize the most important patterns in an input text, we encode its using the state-of-art pre-trained transformer encoder. We compare four transformer encoders, namely BERT, BERT-Large, RoBERTa and RoBERTa-Large (Devlin et al., 2019; Liu et al., 2019).

The tokenizer of the encoder splits the input sentence into wordpeices  $[T_1, T_2, \dots, T_n]$  and encodes them using its vocabulary. The transformer encoder is fed with the encoded input and outputs the pooled embedding  $h_{pooled} \in \mathbb{R}^{1 \times d}$  (embedding of  $[CLS]$  (resp.  $< s >$ ) token of BERT (resp. roBBERTa)) and the CWE  $H = [h_1, h_2, \dots, h_n] \in \mathbb{R}^{n \times d}$  ( $d$  is the embedding dimension).

### 3.2 Task-specific attention layer

We use one task-specific attention layer for each task. Using  $H$ , the CWE of the input sentence, the attention mechanism (Bahdanau et al., 2015; Yang et al., 2016) extracts the task-specific representation  $s_*$  ( $*$  denotes the task) as follows:

$$\begin{aligned} U &= \tanh(W_a H) \\ \alpha &= \text{softmax}(U^T W_\alpha) \\ s_* &= \alpha \cdot H^T \end{aligned}$$

where  $W_a \in \mathbb{R}^{d \times 1}$  and  $W_\alpha \in \mathbb{R}^{n \times n}$  are the trainable parameters of the attention layer,  $U \in \mathbb{R}^{n \times 1}$  is the attention mechanism’s context vector, and  $\alpha \in [0, 1]^n$  weights  $h_1, h_2, \dots, h_n$  according to their contribution to the task objective.

### 3.3 Task Classification/Regression module

As the SemEval-2021 Task-7 consists of two classification tasks (1a and 1c) and two regression tasks (1b and 2), we employ two classification modules and two regression modules. Each of these task-specific module is composed of one hidden layer and one output layer, and takes as input the concatenation  $[h_{pooled}, s_*]$  of the pooled output ( $h_{pooled}$ ) and its task-specific attention output ( $s_*$ ).

### 3.4 MTL objective

Our MTL model is trained to minimize the losses of the four tasks. Specifically, it minimizes the binary cross-entropy loss and the RMSE loss for classification and regression tasks, respectively. These losses are expressed as follows:

- Binary cross-entropy loss for humor classification

$$L_1(\hat{y}, y) = - \sum_{i=1}^N \sum_{j=1}^2 y_i^j \log(\hat{y}_i^j)$$

- RMSE loss for Humor rating

$$L_2(\hat{y}, y) = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

- Binary cross-entropy loss for Controversy classification

$$L_3(\hat{y}, y) = - \sum_{i=1}^N \sum_{j=1}^2 y_i^j \log(\hat{y}_i^j)$$

- RMSE loss for offense rating

$$L_4(\hat{y}, y) = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

where  $y$  and  $\hat{y}$  are the ground-truth and the predicted values, respectively. In order to adaptively weight the losses of the four tasks, we combine them using MTL uncertainty loss weighting (Kendall et al., 2017), given by:

$$\begin{aligned} L_{total}(\hat{y}, y) &= \frac{1}{2\sigma_1^2} L_1(\hat{y}, y) + \frac{1}{2\sigma_2^2} L_2(\hat{y}, y) \\ &+ \frac{1}{2\sigma_3^2} L_3(\hat{y}, y) + \frac{1}{2\sigma_4^2} L_4(\hat{y}, y) \\ &+ \log(\sigma_1 \sigma_2 \sigma_3 \sigma_4) \end{aligned}$$

where  $\sigma_i$  ( $i = 1..4$ ) captures the amount of noise that exists in the output of each task, and used to tune the impact of each loss in MTL optimization. Finally, the MTL model is trained to minimize the overall loss  $L_{total}$  with respect to the network parameters as well as the noise parameters  $\sigma_i$ .

## 4 Results

### 4.1 Experiment Settings

We have evaluated the performance of our model and its single-task counterparts using both the *base* and the *large* models of BERT and RoBERTa:

- **BERT-base**: 12 transformer blocks,  $d = 768$ , 12 attention heads, and 110M parameters.
- **BERT-large**: 24 transformer blocks,  $d = 1024$ , 16 attention heads, and 336M parameters.
- **RoBERTa-base**: 12 transformer blocks,  $d = 768$ , 12 attention heads, and 125M parameters.
- **RoBERTa-large**: 24 transformer blocks,  $d = 1024$ , 16 attention heads, and 355M parameters.

For text preprocessing, we have implemented a simple pipeline that normalizes contractions. All evaluated models are trained using Adam optimizer (Kingma and Ba, 2014) with a learning rate of  $1 \times 10^{-5}$ . The batch size and the number of epochs are fixed to 16 and 5, respectively. We have investigated both single-task training and MTL for all tasks. It is worth mentioning that, for single-task learning, we also apply an attention layer on top of the contextualized word embedding. This has improved single-task models as well. All models are trained on the full train sets, validated on the validation set, and evaluated on the test set of each task. For evaluation purpose, we have used the shared task’s evaluation metrics, namely the **F1-score**, the **Accuracy**, and the Root Mean Squared Error **RMSE**. It is worth mentioning that models’ validation is performed using the development set, while the presented results are obtained employing the test set.

### 4.2 Experiment Results

Table 1 presents the obtained results for all tasks using single-task and MTL models. The results show that our MTL model surpasses its single-task counterparts on all tasks. The large variants of BERT and RoBERTa encoders offer better performance compared to their base variants. The best performance is obtained using our MTL model on top of RoBERTa large encoder. These results can be explained by the fact that deep encoders can capture more complex pattern from the input text.

		Task 1a		Task 1c		Task 1b	Task 2
		Accuracy	F1-score	Accuracy	F1-score	RMSE	RMSE
<b>Single-Task</b>	BERT	0.9171	0.9318	0.4216	0.6023	0.5852	0.5649
	BERT-large	0.9372	0.9425	0.4296	0.6122	0.5748	0.5571
	RoBERTa	0.9408	0.9433	0.4302	0.6035	0.5711	0.5556
	RoBERTa-large	0.9422	0.9531	0.4415	0.6183	0.5688	0.5079
<b>MTL</b>	BERT	0.93	0.9361	0.4296	0.6032	0.5756	0.5511
	BERT-large	0.9421	0.9442	0.4316	0.6054	0.552	0.5533
	RoBERTa	0.945	0.9587	0.4423	0.6152	0.5578	0.5218
	RoBERTa-large <sup>‡</sup>	<b>0.951</b>	<b>0.9606</b>	<b>0.4537</b>	<b>0.6242</b>	<b>0.5401</b>	<b>0.4696</b>

Table 1: The obtained results using our MTL model and its single-task counterparts, with different encoders. The subscript ‡ denotes our official submission to SemEval-2021 Task-7. Results are obtained employing the test set.

		Task 1a		Task 1c		Task 1b	Task 2
		Accuracy	F-score	Accuracy	F-score	RMSE	RMSE
<b>w/o task-attention</b>		0.9335	0.9472	0.4456	0.6122	0.5636	0.4891
<b>w/o uncertainty loss weighting</b>		0.9498	0.9582	0.4516	0.6198	0.5516	0.4722
<b>MTL RoBERTa-large</b>		<b>0.951</b>	<b>0.9606</b>	<b>0.4537</b>	<b>0.6242</b>	<b>0.5401</b>	<b>0.4696</b>

Table 2: Ablation study of our MTL model using MTL RoBERTa-large encoder (w/o denotes without the corresponding component). Results are obtained using the test set.

Besides, MTL leverages useful signals from the related tasks.

To investigate the effectiveness of the task-specific attention layers and the uncertainty loss weighting on the performance of our MTL model, we have performed an ablation study. Table 2 presents the results of our model without these components. The results show that both components improve the performance of our MTL model. We achieve the most performance gain by incorporating the task-specific attention layers into our model. Besides, the adaptive losses weighting component outperforms the simple combination of task losses ( $L_{total} = l_1 + l_2 + l_3 + l_4$ ).

## 5 Conclusion

In this paper, we have presented our system for humor and offense detection and rating. Our system consists of an end-to-end MTL model based on the state-of-art pre-trained transformer encoder and task-specific attention layers. The latter layers are applied on top of the contextualized word embedding to extract task-discriminative features. We have employed two classification and regression modules to tackle the four tasks. Our MTL model is trained to minimize the four tasks losses,

while weighting them adaptively using the MTL uncertainty loss weighting. We have also investigated the performance of our MTL model as well as its single-task counterparts using four pre-trained transformer-based encoders. The best performances are obtained using our MTL model while employing RoBERTa-large encoder.

In future work, we would like to improve our model, by considering the relationship between the different tasks. Besides, we want to use our model not only to detect humorous and offensive content, but also to perform other related tasks.

## References

- Issa Annamoradnejad. 2020. [Colbert: Using BERT sentence embedding for humor detection](#). *CoRR*, abs/2004.12765.
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.
- Sven van den Beukel and Lora Aroyo. 2018. [Homonym detection for humor recognition in short text](#). In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 286–291, Brussels,

- Belgium. Association for Computational Linguistics.
- Andrew Cattle and Xiaojuan Ma. 2018. [Recognizing humour using word associations and humour anchor extraction](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1849–1858, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Cole Gleason, Amy Pavel, Xingyu Liu, Patrick Carrington, Lydia B. Chilton, and Jeffrey P. Bigham. 2019. [Making memes accessible](#). In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility, ASSETS '19*, page 367–376, New York, NY, USA. Association for Computing Machinery.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. [SemEval-2020 task 7: Assessing humor in edited news headlines](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online). International Committee for Computational Linguistics.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. [Multi-task learning using uncertainty to weigh losses for scene geometry and semantics](#). *CoRR*, abs/1705.07115.
- Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.
- Ritesh Kumar, Aishwarya N. Reganti, Akshit Bhatia, and Tushar Maheshwari. 2018. [Aggression-annotated corpus of Hindi-English code-mixed data](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. Semeval-2017 task 6:# hashtagwars: Learning a sense of humor. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57.
- Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. [From humor recognition to irony detection: The figurative language of social media](#). *Data Knowledge Engineering*, 74:1–12. Applications of Natural Language to Information Systems.
- Julian Risch, Robin Ruff, and Ralf Krestel. 2020. [Offensive language detection explained](#). In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 137–143, Marseille, France. European Language Resources Association (ELRA).
- Pranaydeep Singh, Nina Bauwelinck, and Els Lefever. 2020. [Lt3 at semeval-2020 task 8 : multi-modal multi-task learning for memotion analysis](#). In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval 2020)*, pages 1155–1162. Association for Computational Linguistics (ACL).
- Orion Weller and Kevin Seppi. 2019. [Humor detection: A transformer gets the last laugh](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China. Association for Computational Linguistics.
- Michael Wiegand, Melanie Siegel, and Josef Ruppenhofer. 2018. Overview of the germeval 2018 shared task on the identification of offensive language. In *14th Conference on Natural Language Processing KONVENS 2018*.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. [Hierarchical attention networks for document classification](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.
- Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffenseEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(OffenseEval 2020\)](#). In *Proceedings of the Fourteenth*

*Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

Dongyu Zhang, Heting Zhang, Xikai Liu, LIN Hongfei, and Feng Xia. 2019. Telling the whole story: A manually annotated chinese dataset for the analysis of humor in jokes. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6403–6408.



# hub at SemEval-2021 Task 7: Fusion of ALBERT and Word Frequency Information Detecting and Rating Humor and Offense

**Bo Huang**

School of Information Science  
and Engineering Yunnan University,  
Yunnan, P.R. China  
hublucashb@gmail.com

**Yang Bai**

School of Information Science  
and Engineering Yunnan University,  
Yunnan, P.R. China  
baiyang.top@gmail.com

## Abstract

This paper introduces the system description of the hub team, which explains the related work and experimental results of our team's participation in SemEval 2021 Task 7: Ha-Hackathon: Detecting and Rating Humor and Offense. We successfully submitted the test set prediction results of the two subtasks in the task. The goal of the task is to perform humor detection, grade evaluation, and offensive evaluation on each English text data in the data set. Tasks can be divided into two types of subtasks. One is a text classification task, and the other is a text regression task. What we need to do is to use our method to detect the humor and offensive information of the sentence as accurately as possible. The methods used in the results submitted by our team are mainly composed of ALBERT, CNN, and Tf-Idf algorithms. The result evaluation indicators submitted by the classification task are F1 score and Accuracy. The result evaluation index of the regression task submission is the RMSE. The final scores of the prediction results of the two subtask test sets submitted by our team are task1a 0.921 (F1), task1a 0.9364 (Accuracy), task1b 0.6288 (RMSE), task1c 0.5333 (F1), task1c 0.0.5591 (Accuracy), and task2 0.5027 (RMSE) respectively.

## 1 Introduction and Background

Perceiving humor has always been a unique ability of human beings. So what is the use of humor? The research results of Martin and Kuiper on humor show us the influence of humor on a person's physical and mental health (Martin, 2004; Kuiper et al., 2004). In recent years, the use of automated methods to detect humorous information in text has attracted widespread attention (Barbieri and Saggion, 2014; Reyes et al., 2012).

SemEval 2021 Task 7: Ha-Hackathon: Detecting and Rating Humor and Offense's task goal is to

use automated techniques and methods to automatically detect humor and grade in the text. Besides, this task also needs to evaluate the offensive level of the text data. The task is divided into two parts, one part is the detection and evaluation related to humor. There are three subtasks in this part of the task. It involves text classification and regression. The other part is to assess the offensive level of the text data. This is a separate text regression task. The purpose is to predict how offensive the text will be to ordinary users. This task is an interesting challenge for the machine. Humor is a very subjective emotion. People with different cultural backgrounds and life experiences have different feelings about the same sentence. This task is to detect and score humor on the English data set. There are similar tasks and studies in other languages, such as humor scores on Spanish data from tweets (Castro et al., 2018; Chiruzzo et al., 2019).

In text detection and classification tasks, semi-supervised and supervised methods are widely used. Davidov et al. use a semi-supervised method to detect text from social media. The purpose is to detect whether the text data contains ironic information (Davidov et al., 2010). But these methods alone are not enough to make humor ratings on text data. We need to combine semantic information in context. The ELMo (Peters et al., 2018) method based on LSTM (Olah, 2015) overcomes the difficulty that the model cannot learn the context. In the follow-up work, an improved method of ELMo feature extractor appeared. The BERT (Devlin et al., 2018) model based on Transformer Encoder (Vaswani et al., 2017) achieved the best results in many NLP tasks.

## 2 Data and Methods

In this section, we will introduce the data we use in the task and the models and methods we use.

ID	Text	Is humor	Humor rating	Humor controversy	Offense rating
35	Learn from the scars of others	0	0	0	0.05
119	What do you call a sad terrorist? A crisis	1	2.16	1	0.85

Table 1: The training set sample data we used in the task.



Figure 1: The word cloud diagram of the training and test data provided by the task organizer team. The result shown in the figure is the data after removing the stop words.

## 2.1 Data Description

The task organizer team provides each team with training data sets, validation data sets and test data sets related to the “Detecting and Rating Humor and Offense” task (Meaney et al., 2021). We analyze the structure and characteristics of the data sets. The training data set includes ID, Text, Is Humor, Humor Rating, Humor Controversy, Offense Rating. Among them, Is Humor, Humor Rating, Humor Controversy 3 tags are the three subtasks a, b, and c of task 1. Is Humor and Humor Controversy are two binary classification labels, consisting of 0 or 1. Humor Rating is a continuous value between 0-5. Offense Rating is the label of task 2. It is a continuous value between 0-5. The sentence length in the Text is different. Compared with the training data set, the test set only contains the above ID and Text parts. During the development phase, the task organizer also provides a test set. But we did not use this test set in our system, so we do not analyze the test set. We need to use our method to predict the values of Is Humor, Humor Rating, Humor Controversy and Offense Rating labels in the test set. Table 1 shows a sample of the data we used in the task.

8000 and 1000 different sample data constitute the training set and the validation set. The numbers of labels belonging to 1 and 0 in the training set Is Humor label are 4932 and 3068, respectively.

The numbers of labels belonging to 1 and 0 in the training set Humor Controversy label are 2465 and 5535, respectively. The test set consists of 1000 different sample data. We use word cloud graphs to visualize text data. Text data in the training set and test set. The word cloud image clearly shows us the characteristics of word frequency distribution in the text data set. The figure shows the text data after the stop words are deleted. Figure 1 shows word frequency information in the training set and the test set.

## 2.2 Methods

Combined with the analysis and understanding of task description and task data set, we chose to develop an artificial neural network system based on the combination of ALBERT, Tf-Idf and CNN. We also tried to use the combination of BERT and Tf-Idf to verify its impact on the verification set. Both BERT and ALBERT (Lan et al., 2019) are pre-trained language models that are implemented based on the ideas and structure of Transformer. Compared with BERT, ALBERT not only has fewer parameters but also has the characteristics of parameter sharing between different layers. The pre-trained language model also occupies less memory space. Therefore, ALBERT is better than BERT in training effect. The CNN block we use in the system is mainly composed of two-dimensional

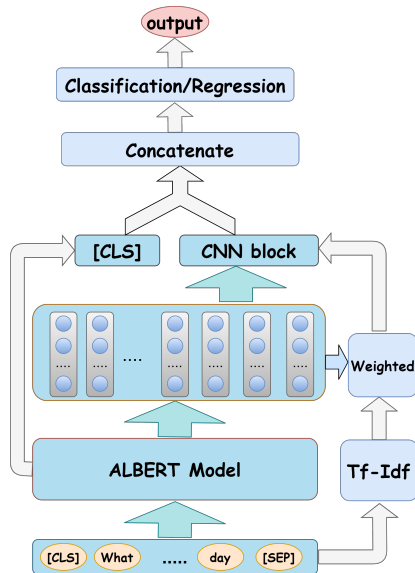


Figure 2: The model structure and data flow we used in the task.

convolution and two-dimensional maximum pooling. The convolution kernel has three (3, 4, 5) different sizes. The processed results of three convolution kernels of different sizes are connected as the output result of the CNN block.

In the system we use to predict the results of the test set. The first step is to input the preprocessed text data into the ALBERT model. At the same time, the text is processed with Tf-Idf to get Tf-Idf\_output. In the ALBERT model, we will get two output values. One is [CLS] (the shape is [batch\_size, hidden\_size]) that contains the entire sentence information. The other is the last layer output of the ALBERT model last\_layer\_output (the shape is [batch\_size, seq\_length, hidden\_size]). In the second step, we use Tf-Idf\_output to perform a weighted operation on last\_layer\_output to get weighted\_output (the shape is [batch\_size, seq\_length, hidden\_size]). In the third step, we use weighted\_output and last\_layer\_output respectively as the input of the CNN block. The two output results have the same shape as [CLS]. In the fourth step, we stitch together the results of the two CNN blocks obtained in the previous step and the results of [CLS] to obtain Concatenate\_output (the shape is [batch\_size, hidden\_size\*3]). In the fifth step, we use Concatenate\_output as the input value of the classifier for the classification or regression task to obtain the predicted output result. Figure 2 shows our model structure and data flow.

### 3 Experiment and Results

In this section, we will introduce the data preprocessing methods and experimental settings we used in the task and the final results.

#### 3.1 Data Preprocessing

Combining our understanding of tasks and data, we removed the stop words in the text data. For the stop word list, we use the stop word package provided by NLTK. Besides, to use the Tf-Idf algorithm to obtain a weighted output, and to ensure that the shape of the text code processed by the Tf-Idf algorithm is consistent with the shape of the ALBERT output, we removed the text code that exceeded the maximum sentence length. For those texts that are less than the maximum sentence length. For text encoding, we perform zero padding.

The validation set we get is unlabeled data, so the validation set we use in the training phase comes from part of the data in the training set. Randomly extract 20% from the training set as the validation set data we will use next.

#### 3.2 Experiment setting

To test the influence of different systems on the prediction results of the task data set. We design several different models and observe the result scores of different models on the validation set. We adjust the parameters as much as possible to obtain the best results for each different model, so different models use different parameter combination settings.

The difference between BERT+Tf-Idf+CNN and the system we introduced above is only to replace the ALBERT model. BERT+Tf-Idf and ALBERT+Tf-Idf directly splice the output results of BERT and ALBERT with the weighted result of Tf-Idf. The other parts are the same as we introduced in section 2.2. BERT and ALBERT directly use their [CLS] output results. The task of classification uses the BCEWithLogitsLoss loss function. The regression task uses the MSELoss loss function.

- ALBERT+Tf-Idf+CNN: The epoch, batch size, maximum sequence length, and learning rate for the model are 5, 32, 70, and  $3e-5$ , respectively.
- BERT+Tf-Idf+CNN: The epoch, batch size, maximum sequence length, and learning rate

Method	task1a	task1	task1b	task1c	task1c	task2
	F1	Acc	RMSE	F1	Acc	RMSE
ALBERT+Tf-Idf+CNN	<b>0.945</b>	0.943	<b>0.605</b>	0.522	<b>0.574</b>	<b>0.492</b>
BERT+Tf-Idf+CNN	0.930	<b>0.944</b>	0.620	<b>0.532</b>	0.561	0.502
ALBERT+Tf-Idf	0.921	0.929	0.617	0.545	0.554	0.490
BERT+Tf-Idf	0.925	0.932	0.627	0.528	0.564	0.491
ALBERT	0.915	0.927	0.634	0.532	0.544	0.510
BERT	0.917	0.923	0.625	0.542	0.551	0.497

Table 2: We use different strategies to get the scores on the validation set.

Method	task1a	task1a	task1b	task1c	task1c	task2
	F1	Acc	RMSE	F1	Acc	RMSE
Top1	0.982	0.985	0.496	0.494	0.630	0.412
Top2	0.975	0.980	0.498	0.470	0.628	0.419
Top3	0.960	0.968	0.521	0.470	0.627	0.423
Our team	0.921	0.936	0.629	0.533	0.559	0.503

Table 3: Part of the results in the leaderboard announced by the task organizer team. Among them, the results of Top1-Top3 are the combination of the scores of the top three in each subtask, not the scores of a team. The total number of participating teams in each of the four subtasks is 58, 50, 36, 48.

for the model are 4, 32, 70, and  $4e-5$ , respectively.

- ALBERT+Tf-Idf: The epoch, batch size, maximum sequence length, and learning rate for the model are 5, 32, 70, and  $3e-5$ , respectively.
- BERT+Tf-Idf: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 70, and  $4e-5$ , respectively.
- ALBERT: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 70, and  $3e-5$ , respectively.
- BERT: The epoch, batch size, maximum sequence length, and learning rate for the model are 4, 32, 70, and  $3e-5$ , respectively.

## 4 Results and Analysis

The evaluation indicators announced by the task organizer team in this task are divided into classification tasks and regression tasks. The classification task uses F1 scores and accuracy scores. The regression task uses root mean squared error (RMSE).

We compare the results obtained by several different methods proposed in the experimental part. The scores of different systems on the same validation set are shown in Table 2. We have the following conclusions:

- Conclusion 1: Introducing additional word frequency information as part of the input information of the model will improve the score of our validation set.
- Conclusion 2: The score difference between ALBERT and BERT on our validation set is not large. But in the same parameters and data set, the training time of ALBERT is shorter than BERT.
- Conclusion 3: Adding a CNN block improves the result score. And the result scores of the two methods based on ALBERT and BERT have their advantages.

The ranking of the test set prediction results announced by the task organizer team uses accuracy scores and RMSE scores respectively. Classification tasks are ranked according to accuracy scores. Regression tasks are ranked according to RMSE scores. We finally submitted the test set prediction result score from the ALBERT+Tf-Idf+CNN system. Because its combined result in training time and score is better than the BERT+Tf-Idf+CNN system. The prediction result scores of the test set we submitted can be seen in Table 3. The scores of our system on the test set are lower than the scores of the top three. But the results of the test set prove that our system and method are feasible.

## 5 Conclusion

In this task related to humor detection and offensive evaluation, we propose an artificial neural network system that combines a pre-trained language model with Tf-Idf and CNN. Although our system is only in the middle of the ranking, we still successfully use our system to predict humor and offensive scores. We studied the contributions of 6 different systems and found that the combination of Tf-Idf and CNN improved the prediction scores of BERT and ALBERT. At the same time, the ALBERT-based system is superior to the BERT-based system in terms of time efficiency. In future work, based on the model we use in the task, we can try to fuse other types of word embedding information, and replace the CNN block with an LSTM block or other artificial neural networks.

## References

- Francesco Barbieri and Horacio Saggion. 2014. Automatic detection of irony and humour in twitter. In *ICCC*, pages 155–162.
- Santiago Castro, Luis Chiruzzo, and Aiala Rosá. 2018. Overview of the haha task: Humor analysis based on human annotation at ibereval 2018. In *IberEval@SEPLN*, pages 187–194.
- Luis Chiruzzo, Santiago Castro, Mathias Etcheverry, Diego Garat, Juan José Prada, and Aiala Rosá. 2019. Overview of haha at iberlef 2019: Humor analysis based on human annotation. In *IberLEF@SEPLN*, pages 132–144.
- Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcasm in twitter and amazon. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 107–116.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Nicholas A Kuiper, Melissa Grimshaw, Catherine Leite, and Gillian Kirsh. 2004. Humor is not always the best medicine: Specific components of sense of humor and psychological well-being. *Humor: International Journal of Humor Research*, 17.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Rod A Martin. 2004. Sense of humor and physical health: Theoretical issues, recent findings, and future directions. *Humor*, 17(1-2):1–19.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Christopher Olah. 2015. Understanding LSTM networks.
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.
- Antonio Reyes, Paolo Rosso, and Davide Buscaldi. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

# YoungSheldon at SemEval-2021 Task 7: Fine-tuning Is All You Need

Mayukh Sharma, Ilanthenral Kandasamy, W.B. Vasantha

School of Computer Science and Engineering

Vellore Institute of Technology

Vellore, Tamil Nadu, India

04mayukh@gmail.com, ilanthenral.k@vit.ac.in,

vasantha.wb@vit.ac.in

## Abstract

In this paper, we describe our system used for SemEval 2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense. We used a simple fine-tuning approach using different Pre-trained Language Models (PLMs) to evaluate their performance for humor and offense detection. For regression tasks, we averaged the scores of different models leading to better performance than the original models. We participated in all SubTasks. Our best performing system was ranked 4 in SubTask 1-b, 8 in SubTask 1-c, 12 in SubTask 2, and performed well in SubTask 1-a. We further show comprehensive results using different pre-trained language models which will help as baselines for future work.

## 1 Introduction

Humor is an intelligent form of communication with the capability of providing amusement and provoking laughter (Chen and Soo, 2018). It helps in bridging the gap between various languages, cultures, and demographics. Humor is a very subjective phenomenon. It can have different intensities, and people may find some jokes funnier than others. In certain situations, some jokes may be offensive to a certain group of people. All these characteristics of humor pose an interesting linguistic challenge to NLP systems. SemEval 2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense (Meaney et al., 2021) aims to draw attention to these challenges in humor detection. The task provides a dataset of humorous content annotated using people representing different age groups, gender, political stances, and income levels. The content of the provided dataset was in English.

Participating in all SubTasks, we propose a fine-tuning based approach on pre-trained language models. Pre-trained Language Models learn syntactic and semantic representations by training on

large amounts of unsupervised data. Recently there has been a lot of interest in PLMs. Researchers have come up with different pre-training methods using Auto Encoding(AE) and Auto-Regressive(AR) language modeling techniques. Often these pre-trained models contain millions of parameters and are computationally expensive. Fine-tuning different models may lead to different results on downstream tasks. This makes the choice of PLM an important factor. We present a comparative study of different PLM models and their performance in all SubTasks of SemEval 2021 Task 7: HaHackathon.

Our proposed fine-tuning approach for each PLM made use of a single layer of one neuron stacked on the PLM features. We performed experiments using BERT(Devlin et al., 2019), ELECTRA(Clark et al., 2020), RoBERTa(Liu et al., 2019), XLNet(Yang et al., 2019), MPNet(Song et al., 2020), and ALBERT(Lan et al., 2020). For regression tasks, we also used averaging technique, which we describe later in the paper. Our model performed well in SubTask 1-b and 1-c, achieving a rank of 4 and 8 respectively on the official leaderboard. For SubTask 2, our proposed averaging technique outperformed individual fine-tuned models by a good margin and was ranked 12. Our code is available online<sup>1</sup> for method replicability.

## 2 Background

The task of automatic humor recognition refers to deciding whether a given sentence expresses humor. The problem of humor recognition is often formulated as a binary classification problem aiming to identify if the given text is humorous. (Weller and Seppi, 2019) performed a study to identify if a joke is humorous or not using transformers

<sup>1</sup><https://github.com/04mayukh/YoungSheldon-at-SemEval-2021-Task-7-HaHackathon>

(Vaswani et al., 2017). They used the body of the joke, the punchline exclusively, and both parts together. Combining both parts lead to better performance and it was found that a punchline carries more weight than the body of a joke for humor identification. (de Oliveira and Rodrigo, 2015) experimented with SVM’s, RNN’s, and CNN’s for identifying humor in Yelp reviews using a bag of words and mean word vector representations. (Chen and Soo, 2018) used CNN-based models for identifying humor content. (Annamoradnejad, 2021) uses a neural network built on BERT embeddings learning features for sentences and whole text separately and then combining them for prediction on 200k Short Texts for Humor Detection dataset on Kaggle<sup>2</sup>. Binary classification tasks help us to separate humorous content but are unable to quantify the degree of humor. SemEval-2017 Task 6: #HashtagWars (Potash et al., 2017) aimed to study the relative humor content of funny tweets by either generating the correct pairwise comparisons of tweets (SubTask A) or finding the correct ranking of the tweets (SubTask B) based on their degree of humor content. SemEval-2020 Task 7: Assessing Humor in Edited News Headlines (Hossain et al., 2020) presented a study on editing news headlines to make them humorous. The task involved quantifying the humor of the edited headline on a scale of (0-3) as well as comparing the humor content of the original and edited headline. SemEval-2020 Task 8: Memotion Analysis- The Visuo-Lingual Metaphor! (Sharma et al., 2020) provides details on humor classification as well as predicting its semantic scale on internet memes using both images and text. OffensEval (Zampieri et al., 2020) (Zampieri et al., 2019) provides insights for identifying offensive content on social media.

Humor is an intelligent way of communication in our daily lives. It helps bridge the gap between people from various cultures, ages, gender, languages, and socioeconomic status making it a powerful tool to connect with the audience. Humor is a highly subjective phenomenon. People from different demographics may have a different perception of humor, and some may even find it offensive. This makes identifying humor a tough task. SemEval 2021 Task 7: Hahackathon: Linking humor and offense across different age groups aims to study this subjective nature of humor, which has two Sub-

<sup>2</sup><https://www.kaggle.com/moradnejad/200k-short-texts-for-humor-detection>

Tasks which we describe as:

SubTask 1: Given a labeled dataset D of texts, the task aims to learn a function that can:

- SubTask 1-a: predict if a text is humorous or not.
- SubTask 1-b: quantify humor present in a humorous text within a range of (0-5).
- SubTask 1-c: predict if the humor rating would be controversial for a humorous text, i.e., the variance of the rating between annotators is higher than the median.

SubTask 2: Given a labeled dataset D of texts, the task aims to learn a regression function that can quantify how offensive a text is for general users within a range of (0-5).

*Dataset Statistics:* Table 1 represents the dataset statistics for classification tasks. For SubTask 1-a we can see there is a slight class imbalance between humorous and non-humorous labels. We overcome the problem of class imbalance using class weights which we define as: Let  $X$  be the vector containing counts of each class  $X_i$  where  $i \in X$ . Then the weights for each class were given as:

$$weight_i = \frac{max(X)}{X_i + max(X)}$$

For SubTask 1-c the label distribution was balanced. Table 2 represents the statistics for the regression tasks. Another observation on the training set for SubTask 2 was that 3388 samples had an offensive rating of 0 and nearly 80% of samples had offense rating in the range 0-1.

### 3 System Overview

#### 3.1 Pre-trained Language Models

NLP being a diverse field contains many tasks, but most task-specific datasets contain only a few hundred or a thousand human-labeled samples. To overcome this problem, researchers have come up with a method called pre-training (Qiu et al., 2020) which involves training general-purpose language representation using enormous amounts of unannotated textual data. These language models can then be fine-tuned on various downstream tasks and have shown promising results in many natural processing tasks (Dai and Le, 2015; Peters et al., 2018; Radford and Narasimhan, 2018). Next, we briefly discuss some pre-trained language models we used for the task.

		Humorous	Non-humorous	Total
Task 1-a	Train	4932	3068	8000
	Validation	632	368	1000
	Test	615	385	1000
		Controversial	Non-controversial	Total
Task 1-c	Train	2465	2467	4932
	Validation	308	324	632
	Test	279	336	615

Table 1: Dataset statistics for classification tasks.

		Mean	Standard Deviation	Count
Task 1-b	Train	2.260	0.566	4932
	Validation	2.269	0.572	632
	Test	2.119	0.546	615
		Mean	Standard Deviation	Count
Task 2	Train	1.393	1.185	8000
	Validation	0.706	1.190	1000
	Test	0.480	0.830	1000

Table 2: Dataset statistics for regression tasks.

### 3.2 Brief overview of used Pre-trained Models

BERT: Bidirectional Encoder Representations from Transformers is a bi-directional language model that uses Transformer (Vaswani et al., 2017) architecture to learn contextual relations between different words in a text sequence (Devlin et al., 2019). It makes use of two training strategies i.e., Masked Language Modelling (MLM) and Next Sentence Prediction (NSP).

ELECTRA: It introduces a new pre-training objective called Replaced Token Detection (RTD) (Clark et al., 2020). Unlike BERT which introduces <MASK> tokens, Electra replaces certain tokens with plausible fakes. The pre-training task then requires the model to determine if the input tokens are the same or have been replaced. This binary classification task is applied to all tokens unlike the small number of masked tokens making RTD more efficient than MLM.

RoBERTa: A Robustly Optimized BERT Pre-training Approach (Liu et al., 2019) was developed by Facebook. They made use of the BERT architecture with modifications to improve the performance on downstream tasks. They made use of dynamic masking in the pre-training objective and removed

the NSP objective. They also trained the model for a longer duration with more data and a larger batch size. They outperformed BERT on several downstream tasks.

XLNet: XLNet (Yang et al., 2019) is a generalized autoregressive pre-training method that takes the best of both AR language modeling and AE modeling techniques. It proposed a permutation language modeling objective for pre-training that helps learn bidirectional contexts. It also helps overcome the pretrain-finetune (Yang et al., 2019) discrepancy present in BERT due to its autoregressive formulation.

MPNet: MPNet (Song et al., 2020) was proposed by Microsoft. It overcomes the positional discrepancy between pre-training and fine-tuning in XLNet which does not use the full position information of a sentence. It proposes a unified view of masked language modeling and permuted language modeling by rearranging and splitting the tokens into predicted and non-predicted parts. It uses MLM and PLM to model the dependency among predicted tokens and see the position information of the full sentence.

ALBERT: A Lite BERT for self-supervised learning of language representations (Lan et al., 2020) is a modification of BERT aiming to effi-



ciently allocate the model capacity to help reduce training time and reduce memory consumption. ALBERT decomposes the embedding matrix into a lower dimension which is then projected to the hidden space. This is called factorized embedding parameterization and helps in reducing the parameters. It also makes use of layer sharing across all layers which helps remove redundancy. Additionally, it uses inter-sentence coherence loss based on Sentence Order Prediction (SOP) (Lan et al., 2020).

### 3.3 Fine-tuning

We fine-tuned the pre-trained language models for each SubTask by stacking a dropout layer followed by a single neuron dense layer on top of PLM features. We used the features of [CLS] token in the case of ALBERT, BERT, XLNet, ELECTRA, and start token (<s>) features in the case of RoBERTa, and MPNet. For the classification task, sigmoid activation was used in the final layer. For the regression task, we did not use any activation. Negative values were converted to zero in regression tasks.

### 3.4 Averaging for Regression tasks

For regression tasks, we combined all fine-tuned models by averaging their predictions. For SubTask 1-b, we averaged the predictions of all models. For SubTask 2 as stated earlier, there were many zero values in the training set therefore, we averaged the predictions only when all models predicted a non-zero value. If any of the models predicted zero for a given sample, we took zero as the final prediction.

## 4 Experimental Setup

We used ekphrasis (Baziotis et al., 2017) library for pre-processing the text inputs. It normalized date, time, numbers to a standard format and also performed spelling correction. For tokenization, we used Hugging Face’s (Wolf et al., 2020) implementation of fast tokenizers for each pre-trained model. We fixed the sequence length of samples to 150 tokens. Models were developed on Keras<sup>3</sup> (Chollet et al., 2015) using the transformers<sup>4</sup> (Wolf et al., 2020) library by Hugging Face. We used Adam (Kingma and Ba, 2017) optimizer for fine-tuning. Learning rate of 1e-4 was used for ELECTRA. For other models, we experimented with 1e-5, 2e-5, and 3e-5. We used binary cross-entropy loss for classification tasks and logCosh loss for regression

<sup>3</sup><https://keras.io>

<sup>4</sup><https://huggingface.co/transformers>

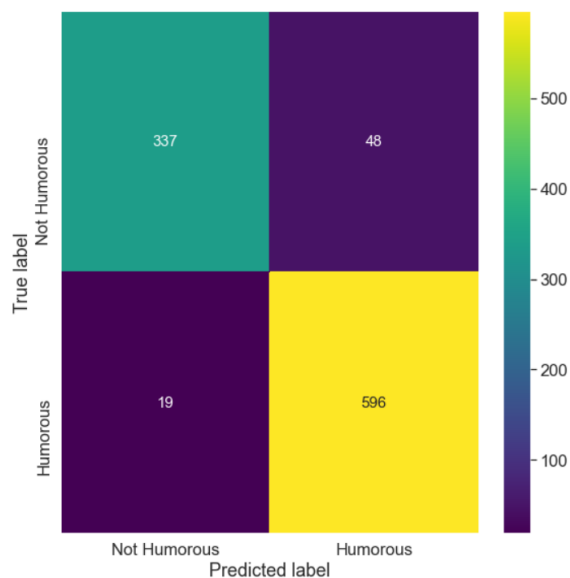


Figure 1: Confusion matrix for SubTask 1-a.

tasks. Batch size of 16 was used for all models. Fine-tuning was performed on TPU’s on Google Colab. We fine-tuned for four epochs on SubTask 1 and 8 epochs on SubTask 2. F1 score and RMSE were used as an evaluation metric for classification and regression tasks. Weights with the best performance on the development set were used for making predictions on the test set.

## 5 Results and analysis

Table 3 shows the results of our proposed fine-tuning approach for different pre-trained models. Our simple averaging technique worked quite well for regression tasks. Our model was ranked 4 in SubTask 1-b and ranked 12 in SubTask 2. The averaging method proposed by us for SubTask 2 provided a significant improvement in the RMSE score against the individually pre-trained models. Upon examination of the test set, we found 40.8% of samples were given zero offense rating. Thus, our decision to predict zero if any of the model predicted zero helped in improving scores against individual models. For classification tasks, our model was ranked 8 in SubTask 1-c and performed well for SubTask 1-a.

Figure 1 and Figure 2 show plots of confusion matrices for our best performing model fine-tuned on BERT. For Subtask 1-a, our model was efficient in separating the humorous and non-humorous content as false positives are low for each class. For SubTask 1-c, our model performed well in identifying the controversial text but did not perform

Model	Precision		Recall		F1		Accuracy	
	1-a	1-c	1-a	1-c	1-a	1-c	1-a	1-c
Subtask								
BERT-base	.9254	.4630	<b>.9691</b>	<b>.9426</b>	<b>.9467</b>	<b>.6210</b>	<b>.9330</b>	.4780
ELECTRA-base	.9269	.4747	.9073	.7419	.9170	.5790	.8990	.5105
RoBERTa-base	.9518	.4708	.8991	.8960	.9247	.6172	.9100	.4959
MPNet-base	<b>.9658</b>	<b>.4879</b>	.9203	.7275	.9425	.5841	.9310	<b>.5300</b>
XLNet-base	.9489	.4812	.9365	.6881	.9427	.5663	.9300	.5219
ALBERT-large	.9632	.4666	.8943	.8530	.9274	.6032	.9140	.4910

Table 3: Test set results for SubTask 1-a and SubTask 1-c.

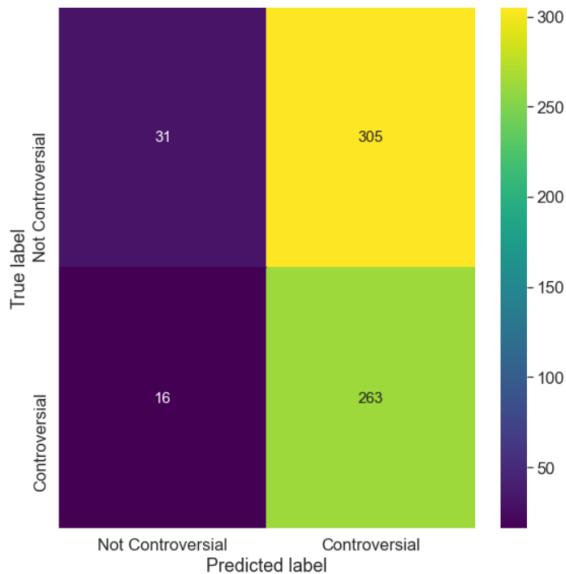


Figure 2: Confusion matrix for SubTask 1-c.

very well for non-controversial text. The model has a very high recall but low precision due to high false positives for the controversial class which is evident from the confusion matrix. BERT, MPNet, and XLNet performed better than other PLMs for SubTask 1-a. For SubTask 1-b individual models had a similar performance. Averaging helped in improving the performance. BERT, ELECTRA, and ALBERT had the best performance on the test set for SubTask 1-c.

## 6 Conclusion

The paper describes our system used for competing in all SubTasks of SemEval 2021 Task 7: Ha-Hackathon: Detecting and Rating Humor and Offense. We used a simple fine-tuning approach for analyzing the performance of various pre-trained language models for the task of humor detection. We performed well in all SubTasks except SubTask 1-a. A lot of research is happening around

Model	RMSE	
	Task 1-b	Task 2
SubTask		
BERT-base	.5380	.5066
ELECTRA-base	.5418	.6071
RoBERTa-base	.5428	.5046
MPNet-base	.5401	.5142
XLNet-base	.5380	.5298
ALBERT-large	.5307	.5004
PLM Average	<b>.5257</b>	<b>.4499</b>

Table 4: Test set results for SubTask 1-b and SubTask 2.

pre-trained language models with new and better models coming up. These models are large and computationally expensive. Choosing a model becomes a difficult task as they may have different results on downstream tasks. We, therefore, performed experiments with the recent state-of-the-art models and provide a comparative analysis of their performance. In the future, we would like to work on the effect of pre-training PLMs with additional task-specific data and then fine-tuning to see their performance on downstream tasks.

## References

- Issa Annamoradnejad. 2021. [Colbert: Using bert sentence embedding for humor detection](#).
- Christos Baziotis, Nikos Pelekis, and Christos Douk-eridis. 2017. Dastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.
- Peng-Yu Chen and Von-Wun Soo. 2018. [Humor recognition using deep learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Pa-*

- pers), pages 113–117, New Orleans, Louisiana. Association for Computational Linguistics.
- François Chollet et al. 2015. Keras. <https://keras.io>.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. **Electra: Pre-training text encoders as discriminators rather than generators**. In *International Conference on Learning Representations*.
- Andrew M Dai and Quoc V Le. 2015. **Semi-supervised sequence learning**. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. **SemEval-2020 task 7: Assessing humor in edited news headlines**. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online). International Committee for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2017. **Adam: A method for stochastic optimization**.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. **Albert: A lite bert for self-supervised learning of language representations**. In *International Conference on Learning Representations*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **Roberta: A robustly optimized bert pretraining approach**.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. **Semeval 2021 task 7, hahackathon, detecting and rating humor and offense**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Luke de Oliveira and A. Rodrigo. 2015. **Humor detection in yelp reviews**.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. **Deep contextualized word representations**. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. **SemEval-2017 task 6: #HashtagWars: Learning a sense of humor**. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57, Vancouver, Canada. Association for Computational Linguistics.
- XiPeng Qiu, TianXiang Sun, YiGe Xu, YunFan Shao, Ning Dai, and XuanJing Huang. 2020. **Pre-trained models for natural language processing: A survey**. *Science China Technological Sciences*, 63(10):1872–1897.
- A. Radford and Karthik Narasimhan. 2018. **Improving language understanding by generative pre-training**.
- Chhavi Sharma, Deepesh Bhageria, William Scott, Srinivas PYKL, Amitava Das, Tanmoy Chakraborty, Viswanath Pulabaigari, and Björn Gambäck. 2020. **SemEval-2020 task 8: Memotion analysis- the visuo-lingual metaphor!** In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 759–773, Barcelona (online). International Committee for Computational Linguistics.
- Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. **Mpnet: Masked and permuted pre-training for language understanding**. In *NeurIPS 2020*. ACM.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. **Attention is all you need**. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Orion Weller and Kevin Seppi. 2019. **Humor detection: A transformer gets the last laugh**. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. **Transformers: State-of-the-art natural language processing**. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. **Xlnet: Generalized autoregressive pretraining for language understanding**. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [SemEval-2019 task 6: Identifying and categorizing offensive language in social media \(OffensEval\)](#). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(OffensEval 2020\)](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

# MagicPai at SemEval-2021 Task 7: Method for Detecting and Rating Humor Based on Multi-Task Adversarial Training

Jian Ma, Shuyi Xie, Haiqin Yang<sup>§</sup>, Lianxin Jiang,  
Mengyuan Zhou, Xiaoyi Ruan, Yang Mo

Ping An Life Insurance, Ltd.

Shenzhen, Guangdong province, China

{MAJIAN446,XIESHUYI542,JIANGLIANXIN769,RUANXIAOYI687,MOYANG853}@pingan.com.cn

<sup>§</sup> the corresponding author, email: hqyang@ieee.org

## Abstract

This paper describes MagicPai’s system for SemEval 2021 Task 7, HaHackathon: Detecting and Rating Humor and Offense. This task aims to detect whether the text is humorous and how humorous it is. There are four subtasks in the competition. In this paper, we mainly present our solution, a multi-task learning model based on adversarial examples, for task 1a and 1b. More specifically, we first vectorize the cleaned dataset and add the perturbation to obtain more robust embedding representations. We then correct the loss via the confidence level. Finally, we perform interactive joint learning on multiple tasks to capture the relationship between whether the text is humorous and how humorous it is. The final result shows the effectiveness of our system.

## 1 Introduction

Humor is the tendency of experiences to provoke laughter and provide amusement. Regardless of gender, age or cultural background, it is a special way of language expression to provide an active atmosphere or resolve embarrassment in life while being an important medium for maintaining mental health (Lefcourt and Martin, 1986). Recently, with the rapid development of artificial intelligence, it becomes one of the most hot research topics in natural language processing to recognize humor (Nijholt et al., 2003). The task of humor recognition consists of two subtasks: whether the text contains humorous and what level of the humor it is. Early humor recognition methods tackle this task mainly by designing heuristic humor-specific features on classification models (Khodak et al., 2018) and have proved that this automatic way can attain satisfactory performance. Nowadays, researchers try to resolve this task by statistical machine learning or deep learning technologies.

The SemEval 2021 Task 7, HaHackathon: Detecting and Rating Humor and Offense, consists of four subtasks: Subtask 1 simulates the previous humor detection task, in which all scores are averaged to provide an average classification score. Subtask 1a is a binary classification task to detect whether the text is humorous. Subtask 1b is a regression task to predict how humorous it is for ordinary users in a value range from 0 to 5. Subtask 1c is also a binary classification task to predict whether the humor grade causes controversy if the text is classified as humorous. Subtask 2 aims to predict how offensive text for an ordinary user is in an integral value range from 0 and 5.

Due to the highly subjective nature of humor detection, the data is labeled by people with different profile in gender, age group, political position, income level, social status, etc. The tasks are extremely challenging because they lack a unified standard to define humor.

To tackle the tasks, we first preprocess the text, including stemming, acronym reduction, etc. We then apply the pre-trained language model to get the representation of each subword in the text as the model input. Meanwhile, we add a perturbation to the embedding layer and design an optimization goal that maximizes the perturbation of the loss function. After that, we perform interactive multi-task learning on judging whether humor exists and predicting how humorous it is. That is, based on maximizing the likelihood estimation under the Gaussian distribution with the same variance, we construct a multi-task loss function and automatically select different loss weights in the learning to improve the accuracy of each task.

## 2 Related Work

The early stages of humor recognition are based on statistical machine learning methods. For example,

Taylor and Mazlack (2004) try to learn statistical patterns of text in N-grams and provide a heuristic focus for a location of where wordplay may or may not occur. Mihalcea and Strapparava (2005) show that automatic classification techniques can be effectively deploy to distinguish between humorous and non-humorous texts and obtain significant improvement over the Apriori algorithm, a well-known baseline. In addition, three human-centric features are designed for recognizing humor in the curated one-liner dataset. Mikolov et al. (2011) apply SVM models for humor recognition as a binary classification task and prove that the technique of metaphorical mapping can be generalized to identify other types of double entendre and other forms of humor. Kiddon and Brun (2011) present several modifications of the original recurrent neural network language model to solve the humor recognition task. Castro et al. (2016) collect a crowd-sourced corpus for humor classification from Spanish tweets and conduct extensive experiments to compare various machine learning models, such as Support Vector Machine (SVM), a Multinomial version of Naïve Bayes (MNB), Decision Trees (DT), k Nearest Neighbors (kNN), and a Gaussian version of Naïve Bayes (GNB). Yan and Pedersen (2017) observe that bigram language models performed slightly better than trigram models and there is some evidence that neural network models can outperform standard back-off N-gram models. Chen and Soo (2018) extend the techniques of automatic humor recognition to different types of humor as well as different languages in both English and Chinese and proposed a deep learning CNN architecture with high way networks that can learn to distinguish between humorous and nonhumorous texts based on a large scale of balanced positive and negative dataset.

With the rapid development of deep learning technology, various pre-training models have made great progress in the field of natural language processing (Yang and Shen, 2021; Wang et al., 2021; Yang et al., 2021). Liu et al. (2018) propose to model sentiment association between elementary discourse units and compare various CNN methods of humor recognition. Weller and Seppi (2019) employ a Transformer architecture for its advantages in learning from sentence context and demonstrate the effectiveness of this approach and show results that are comparable to human performance. Ma et al. (2020) propose a new algorithm Enhance-

ment Inference BERT (EI-BERT) that performs well in sentence classification. Fan et al. (2020) propose an internal and external attention neural network (IEANN) Attention mechanism (Fan et al., 2020; Jiao et al., 2019) has been applied and show good model performance. The existing work can be borrowed or inspired our proposal in this paper.

### 3 Overview

In the following, we present the implementation of our system for the competition.

#### 3.1 Virtual Adversarial Training Based on Loss Correction

Recently, adversarial examples (Szegedy et al., 2014) have been generated to increase the robustness of training deep learning models (Pan et al., 2019; Lei et al., 2020). This work is motivated by the significant discontinuities between the input-output mappings of deep neural networks. When an imperceptible perturbation is added to the input, it may make the original normal network misclassify the result. The characteristics of these perturbations are not random artifacts of learning generated by the network during the learning process, because the same perturbation will cause different networks trained on different data sets to produce the same classification errors. Adversarial examples are samples that significantly improve the loss of the model by adding small perturbations to the input samples.

The adversarial training (Miyato et al., 2017) is a training process that can effectively identify the original sample and the adversarial sample model. Usually, the adversarial training requires labeled samples to provide supervision loss because the perturbation is designed to increase the model loss function. Virtual adversarial training (Liu et al., 2020) extends the adversarial training to semi-supervised mode by adding regularization to the model so that the output distribution of a sample is the same as the output distribution after perturbation while attaining good performance in both supervised and unsupervised tasks. When the training sample is mixed with noise, it is easy to overfit the model and learn wrong information. Therefore, it is necessary to interfere to control the influence of noise.

Figure 1(a) illustrates the perturbation in our implementation. For a word with a sequence length of  $n$ , we let  $w_i$  denote the  $i$ -th subword, where  $i = 1, \dots, n$ . The representation of  $w_i$  is then

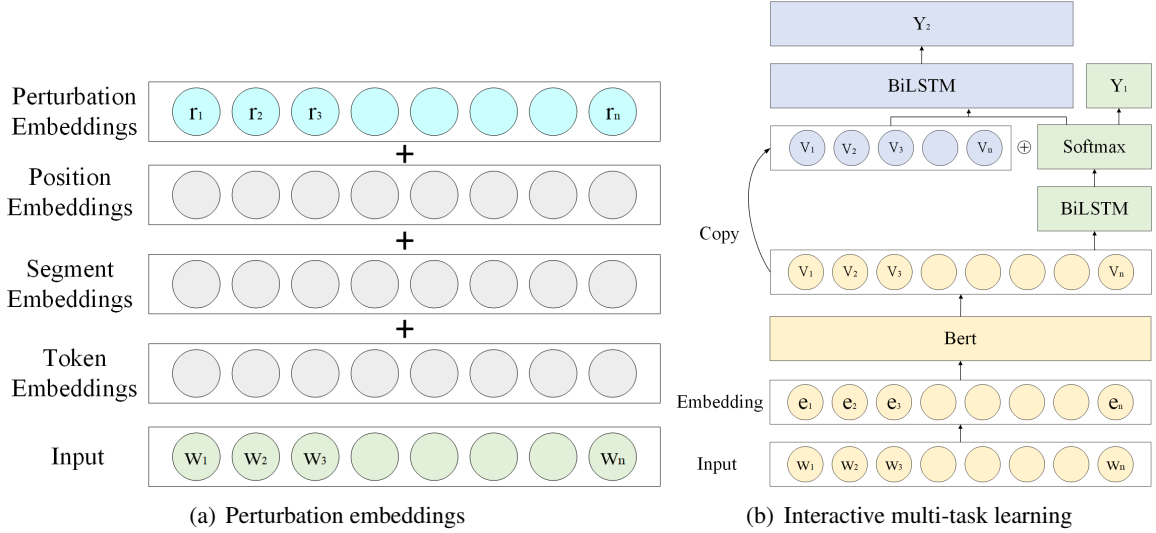


Figure 1: The main implementation of our proposed system.

computed by the sum of token embedding, segmentation embedding, position embedding, and perturbation embedding, an additional embedding. This makes it slightly different from the existing pre-trained language models, e.g., BERT.

The virtual adversarial training can be unified by the following objective:

$$\min_{\theta} \mathbb{E}_{(x,y) \sim D} [\alpha \max_{\beta} L(f(x + \beta; \theta), f(x; \theta)) + L(f(x; \theta), y)], \quad (1)$$

where  $D$  is a training dataset consisting of input-output pairs  $(x, y)$ ,  $\alpha$  is a hyperparameter to control the trade-off between the standard error and the robust error.  $\beta$  is the adversarial perturbation,  $y$  represents the true label,  $\theta$  is the model parameter,  $L$  is the loss function.  $x + \beta$  quantifies the perturbation  $\beta$  injecting into  $x$ . The goal of  $\beta$  is to maximize the difference between the two decision function values,  $f(x + \beta; \theta)$  and  $f(x; \theta)$ , i.e., to make the prediction of the existing model as incorrect as possible. To make  $\beta$  meet a certain constraint, a conventional setting is to let  $\|\beta\| \leq \epsilon$ , where  $\epsilon$  is a constant. After constructing an adversarial sample  $x + \beta$  for each sample, Eq. (1) tries to seek the model parameter  $\theta$  by minimizing the prediction loss.

Since the training samples are mixed with noise, it is easy for the model to overfit and learn wrong information (Reed et al., 2015), interference is adopted to control the influence of noise. The loss

function is defined as follows:

$$\mathcal{L} = - \sum_{i=1}^N ((1 - w_i)y_i + w_i\tilde{y}_i) \log(l_i) \quad (2)$$

where  $y_i$  is the true label,  $\tilde{y}_i$  is the predicted label, and  $l_i$  is the predicted probability distribution.  $w_i$  is a hyperparameter to control the trade-off between true label and predicted label. By minimizing the loss defined in Eq. (2), we can reduce the attention to noise points by adding the model's own predictions to the true labels and the prediction to the noise point.

### 3.2 Interactive Multi-task Training

According to the description of the first two tasks, task 1a is a binary classification task to predict if the text would be considered humorous for an average user while task 1b is a regression task to determine how humorous it is for an average user when the text is classed as humorous, where the values vary between 0 and 5. In order to capture the relationship between whether text is humorous and how humorous it is, we designed the network structure shown in Fig. 1(b). The input, as illustrated in Fig. 1(a), is the sum of the token embedding, position embedding, segment embedding, and perturbation embedding. The sum of four embeddings is sent to a pre-trained language model (PLM) to yield an input for a BiLSTM model. After that, a Softmax layer is placed to recognize whether the text is humor. Meanwhile, the output of the PLM and the output of the Softmax layer are concatenated together and sent to another BiLSTM model

to predict how humorous it is. In Fig. 1(b), the notation  $\oplus$  represents the concatenation operation. Because two tasks have different noise patterns, learning two tasks simultaneously can make features interact in the tasks. For task 1a, it is easy to learn some important features while for task 1b, it is difficult to extract them. The reason may come from the following facts: the interaction between task 1b and the features may be too complicated, or some other features may hinder the learning procedure (Xia and Ding, 2019). Hence, by deploying interactive multi-task learning, we can get a more generalized representation.

Since different loss functions have different scales, loss functions with a larger scale will significantly dominate the loss functions with a smaller scale (Liang et al., 2020; Zhang et al., 2020). Therefore, a weighted summation of the loss function is required to make balance on the loss functions. Motivating by (Kendall et al., 2017) that modeling is based on task-dependent and homoscedastic aleatoric uncertainty, i.e., for a certain sample, the model not only predicting its label but also estimating the task-dependent homoscedastic uncertainty, we present a multi-task loss function derived by maximizing the Gaussian likelihood of the same variance uncertainty. Suppose the input is  $X$ , the parameter matrix  $W$  is the model parameter for the output,  $f^W(x)$ . For the classification in task 1a, the Softmax likelihood can be defined by:

$$p(y_1|f^W(x)) = \text{Softmax}(f^W(x), \sigma_1), \quad (3)$$

where  $\sigma_1$  is the observed noise scalar for the classification model.

For the regression task in task 1b, we can define its probability as the Gaussian likelihood by:

$$p(y_2|f^W(x)) = G(f^W(x), \sigma_2), \quad (4)$$

where  $\sigma_2$  is the observed noise scalar for the regression model.

Here, to learn the models in the multi-task mode, we define the multivariate probability by

$$\begin{aligned} & p(y_1, y_2|f^W(x)) & (5) \\ & = p(y_1|f^W(x)) \cdot p(y_2|f^W(x)) \\ & = \text{Softmax}(f^W(x), \sigma_1) \cdot G(f^W(x), \sigma_2). \end{aligned}$$

Maximizing the probability defined in Eq. (5) is

equivalent to minimizing the following objective:

$$\begin{aligned} & L(W, \sigma_1, \sigma_2) & (6) \\ & = -\log p(y_1, y_2|f^W(x)) \\ & = -\log \text{Softmax}(f^W(x), \sigma_1) \cdot G(f^W(x), \sigma_2) \\ & \propto \frac{1}{\sigma_1^2} L_1(W) + \frac{1}{2\sigma_2^2} L_2(W) + \log \sigma_1 + \log \sigma_2 \end{aligned}$$

where  $L_1 = -\log \text{Softmax}(f^W(x), y_1)$  defines the cross entropy loss between the prediction and  $y_1$ .  $L_2 = \|y_2 - f^W(x)\|^2$  defines the Euclidean loss between the prediction and  $y_2$ . By minimizing the above objective, we can learn the parameters of  $W$ ,  $\sigma_1$ , and  $\sigma_2$  accordingly.

	Train			Dev			Test		
	No.	R.	L.	No.	R.	L.	No.	R.	L.
1a	8000	7:3	20	1000	5:3	19	1000	*	23
1b	4935	*	19	1000	*	19	1000	*	23
1c	4935	1:1	19	1000	1:1	19	1000	*	23
2	8000	*	20	1000	*	19	1000	*	23

Table 1: Data Statistics. R.: The ratio of positive and negative samples. L.: the average length. \* indicates the data is unavailable.

	BERT	RoBERTa	XLNet	ERNIE
lr	2e-5	5e-6	5e-6	3e-5
nfe	5	10	10	15
bs	64	32	32	32
mssl	128	100	80	80
wp	0.05	0.1	0.05	0.05

Table 2: Parameters for different pre-trained language models. lr: learning rate. nfe: no. of training epochs. bs: batch size. mssl: max. sequence length. wp: warmup proportion.

## 4 Experiments

In the following, we present the data, experimental setup and analyze the results.

### 4.1 Data and Experimental Setup

The data is collected from the official release in (Meaney et al., 2021). We preprocess the data by spelling correction, stemming, handling special symbols, and converting all letters to lowercase, etc. Finally, we obtain the data and report the statistics in Table 1.

In the experiment, we choose the large version of four popular pre-training language models, i.e.,



BERT, XLNet, RoBERTa, and ERNIE. The hyperparameters of each model are tuned based on our experience and shown in Table 2. To train a good classifier, we deliver the following procedure: 1) conducting five-fold cross-validation on the training set and obtaining 20 models; 2) applying the 20 models to get the pseudo-labels of the data in the test set and extracting the data with high confidence, i.e., the predicted label score greater than 0.95 or smaller than 0.05, as new training data; 3) the pseudo label data from the test set are mixed with the original training set to train new models. Finally, 892 pseudo label data are selected and mixed with the training set to train the final models. The regression model is jointly trained with the classification models. The models that performed well in cross-validation are selected and averaged by the weighted fusion based on the confidence.

Models	AT	LC	AT + LC
BERT	0.9459	0.9490	<b>0.9534</b>
RoBERTa	0.9480	0.9482	<b>0.9569</b>
XLNet	0.9462	<b>0.9487</b>	0.9470
ERNIE	0.9491	0.9499	<b>0.9512</b>

Table 3: The performance (accuracy) of task 1a with different training strategies.

Models	1a (Acc.)	1a (F1)	1b (RMSE)
ST	0.9569	0.9470	0.6059
MT	0.9577	0.9480	0.5823
MT+WL	<b>0.9637</b>	<b>0.9550</b>	<b>0.5701</b>

Table 4: Comparison of different strategies. ST: single task. MT: multi-task. WL: weigh loss.

## 4.2 Results

In order to prove the effectiveness of adversarial training (AT) and loss correction (LC), we verify task 1a on four pre-training models. AT denotes the models through adversarial training by adding perturbations in the embedding layer. LC denotes the strategy to make correction on the classification cross entropy to interfere with the influence of noise on the model. AT+LC means to apply both strategies in the training. Results reported in Table 4 show that by employing individual strategy, the models can attain good performance on task 1a while employing both strategies can gain better accuracy in BERT, RoBERTa, and ERNIE.

Moreover, we verify the effectiveness of the in-

teractive multi-task training strategy on RoBERTa. MT+WL denotes that the weighted hyperparameters in the loss function are adjusted based on uncertainty, determined by the learned  $\sigma_1$ 's and  $\sigma_2$ , during interactive multi-task training to scale the output the loss function of each task in a similar range. Results reported Table 4 show that the multi-task joint training mechanism can reduce the RMSE of the regression task (i.e., 1b) significantly while adjusting the loss weight can further decrease the error.

Finally, we attain the F1 score of 0.9570 and the accuracy of 0.9653 on task 1a, respectively. The RMSE on task 1b is 0.5572. The RMSE on task 2 is 0.446.

## 5 Conclusion and Future Work

This paper presents our system for SemEval-2021 task 7. Several techniques, such as interactive multi-task joint training, adversarial training, and loss correction, are applied to tackle the task. More specifically, the perturbation is first added to the input embedding layer and the predicted labels are also added with the real labels to reduce the loss of the noise point data. Next, the output of task 1a by the Softmax is concatenated with the input of the task 1b to perform joint training on both tasks. Meanwhile, the uncertainty weighting scheme on the loss allows the simple task to have a higher weight. Finally, multiple models are ensembled to yield the final prediction results. Our system attains the first place in the competition.

In the future, we can explore and verify three other effective strategies. The first strategy is the task-adaptive funetuning on the pre-trained language models. Relevant sentences can be continuously fed into the pre-trained language models to improve the model performance. The second strategy is to build a graph neural network (GNN) model to exploit all vocabulary for text classification. Because BERT is relatively limited to capture the global information from a larger language vocabulary, it is promising to facilitate the GNN, which captures the global information, with the in-depth interaction of BERT's middle layers, which embed sufficient local information. We will further investigate discourse structures (Lei et al., 2017, 2018) for humor detection. Because, both BERT and GNN models information from word relations, it is necessary to involve the study of discourse structures, which describe how two sentences are

logically connected to one another. By such novel design, we can attain better representations and improve the classification performance.

## References

- Santiago Castro, Matías Cubero, Diego Garat, and Guillermo Moncecchi. 2016. [Is this a joke? detecting humor in spanish tweets](#). In *Advances in Artificial Intelligence - IBERAMIA 2016*, pages 139–150, Cham. Springer International Publishing.
- Peng-Yu Chen and Von-Wun Soo. 2018. [Humor recognition using deep learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117, New Orleans, Louisiana. Association for Computational Linguistics.
- Xiaochao Fan, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Yanbo Zou. 2020. [Humor detection via an internal and external neural network](#). *Neurocomputing*, 394:105–111.
- Wenxiang Jiao, Haiqin Yang, Irwin King, and Michael R. Lyu. 2019. [Higr: Hierarchical gated recurrent units for utterance-level emotion recognition](#). In *NAACL-HLT*, pages 397–406. Association for Computational Linguistics.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2017. [Multi-task learning using uncertainty to weigh losses for scene geometry and semantics](#). *CoRR*, abs/1705.07115.
- Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. [A large self-annotated corpus for sarcasm](#). In *LREC*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Chloé Kiddon and Yuriy Brun. 2011. [That’s what she said: Double entendre identification](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 89–94, Portland, Oregon, USA. Association for Computational Linguistics.
- Herbert M. Lefcourt and Rod A. Martin. 1986. *Humor and Life Stress*. Springer New York.
- Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. 2020. [Conversational recommendation: Formulation, methods, and evaluation](#). In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2425–2428.
- Wenqiang Lei, Xuancong Wang, Meichun Liu, Ilija Ilievski, Xiangnan He, and Min-Yen Kan. 2017. [Swim: A simple word interaction model for implicit discourse relation recognition](#). In *IJCAI*, pages 4026–4032.
- Wenqiang Lei, Yuanxin Xiang, Yuwei Wang, Qian Zhong, Meichun Liu, and Min-Yen Kan. 2018. [Linguistic properties matter for implicit discourse relation recognition: Combining semantic interaction, topic continuity and attribution](#). In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Hongru Liang, Wenqiang Lei, Paul Yaozhu Chan, Zhenglu Yang, Maosong Sun, and Tat-Seng Chua. 2020. [Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music](#). In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 574–582.
- Lizhen Liu, Donghai Zhang, and Wei Song. 2018. [Modeling sentiment association in discourse for humor recognition](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 586–591, Melbourne, Australia. Association for Computational Linguistics.
- Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. [Adversarial training for large neural language models](#).
- Jian Ma, ShuYi Xie, Meizhi Jin, Jiang Lianxin, Mo Yang, and Jianping Shen. 2020. [XSYSIGMA at SemEval-2020 task 7: Method for predicting headlines’ humor based on auxiliary sentences with EIBERT](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1077–1084, Barcelona (online). International Committee for Computational Linguistics.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, and Walid Magdy. 2021. [Semeval 2021 task7, ha-hackathon, detecting and rating humor and offense](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Rada Mihalcea and Carlo Strapparava. 2005. [Making computers laugh: Investigations in automatic humor recognition](#). In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 531–538, Vancouver, British Columbia, Canada. Association for Computational Linguistics.
- T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur. 2011. [Extensions of recurrent neural network language model](#). In *ICASSP*, pages 5528–5531.
- Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. [Adversarial training methods for semi-supervised text classification](#). *Machine Learning*.
- Anton Nijholt, Oliviero Stock, Alan Dix, and John Morkes. 2003. [Humor modeling in the interface](#). *Conference on Human Factors in Computing Systems - Proceedings*.

- Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.
- Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training deep neural networks on noisy labels with bootstrapping. *Computer Vision and Pattern Recognition*, abs/1705.07115.
- Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *Computer Vision and Pattern Recognition*.
- Julia M. Taylor and Lawrence J. Mazlack. 2004. Computationally recognizing wordplay in jokes. *Proceedings of the Annual Meeting of the Cognitive Science Society*, 26.
- Xinyi Wang, Haiqin Yang, Liang Zhao, Yang Mo, and Jianping Shen. 2021. Refbert: Compressing bert by referencing to pre-computed representations. In *IJCNN*.
- Orion Weller and Kevin D. Seppi. 2019. Humor detection: A transformer gets the last laugh. *CoRR*, abs/1909.00252.
- Rui Xia and Zixiang Ding. 2019. Emotion-cause pair extraction: A new task to emotion analysis in texts. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1003–1012, Florence, Italy. Association for Computational Linguistics.
- Xinru Yan and Ted Pedersen. 2017. Duluth at semeval-2017 task 6: Language models in humor detection. *CoRR*, abs/1704.08390.
- Haiqin Yang and Jianping Shen. 2021. Emotion dynamics modeling via bert. In *IJCNN*.
- Haiqin Yang, Xiaoyuan Yao, Yiqun Duan, Jianping Shen, Jie Zhong, and Kun Zhang. 2021. Progressive open-domain response generation with multiple controllable attributes. In *IJCAI*.
- Yao Zhang, Xu Zhang, Jun Wang, Hongru Liang, Wenqiang Lei, Zhe Sun, Adam Jatowt, and Zhenglu Yang. 2020. Generalized relation learning with semantic correlation awareness for link prediction. *arXiv preprint arXiv:2012.11957*.

# UPB at SemEval-2021 Task 7: Adversarial Multi-Task Learning for Detecting and Rating Humor and Offense

Răzvan-Alexandru Smădu, Dumitru-Clementin Cercel, Mihai Dascalu

University Politehnica of Bucharest, Faculty of Automatic Control and Computers

razvan.smadu@stud.acs.upb.ro

{dumitru.cercel, mihai.dascalu}@upb.ro

## Abstract

Detecting humor is a challenging task since words might share multiple valences and, depending on the context, the same words can be even used in offensive expressions. Neural network architectures based on Transformer obtain state-of-the-art results on several Natural Language Processing tasks, especially text classification. Adversarial learning, combined with other techniques such as multi-task learning, aids neural models learn the intrinsic properties of data. In this work, we describe our adversarial multi-task network, AML-Humor, used to detect and rate humor and offensive texts from Task 7 at SemEval-2021. Each branch from the model is focused on solving a related task, and consists of a BiLSTM layer followed by Capsule layers, on top of BERTweet used for generating contextualized embeddings. Our best model consists of an ensemble of all tested configurations, and achieves a 95.66% F1-score and 94.70% accuracy for Task 1a, while obtaining RMSE scores of 0.6200 and 0.5318 for Tasks 1b and 2, respectively.

## 1 Introduction

Sentiment analysis studies expressed opinions and feelings, affective states, and subjective information introduced while conveying ideas to others. In recent years, social media has encountered a major change in ways of interaction and in the freedom of expressing moods to others. For example, individuals frequently use emojis, or even abbreviations, to emphasize their feelings. Therefore, new machine learning systems are developed to predict sentiment valences; however, this task is not trivial, as the detection of sentiments is challenging even for humans. This is mainly generated by the diversity of contexts and differences among people, such as socio-cultural status, age, or gender. For example, certain jokes might be confusing due to

past experiences or the current mood while reading the joke. Similarly, language ambiguity can introduce difficulties in understanding the text and in altering the perceived emotions.

Moreover, individuals express a wide range of sentiments, such as happiness, humor, or anger. Words that are used to express such feelings, might share multiple valences and meanings, depending on the context. Therefore, a sequence of words that is, for example, humorous in a certain context, can be considered offensive in another situation. Such examples can rely on bad jokes regarding a person's position in society, ethnicity, or political ideology. Another characteristic that impacts a text's level of humor is how clear the idea is conveyed to the reader.

In this regard, SemEval-2021 Task 7 - HaHackathon: Detecting and Rating Humor and Offense (Meaney et al., 2021) introduced four sub-tasks: *Task 1a* (classification) includes the class of the text, which can be either humorous or not; *Task 1b* (regression) presents the level of humor as a value between 0 and 5; *Task 1c* (classification) covers the rating of the class of being controversial (i.e., some annotators considered a text to be humorous, while others considered the opposite); *Task 2* (regression) contains the level of offensiveness from the text (i.e., how offensive is the text as an aggregate score agreed by all annotators).

Each example in the dataset contains four labels, one for each task. The labels for Tasks 1b and 1c depend on whether the text contains humor or not. Task 2 is independent of Task 1 and all examples from the dataset have assigned scores corresponding to their level of offensiveness.

In this work, we introduce an adversarial multi-task learning (Liu et al., 2017) architecture to detect humor in texts, as well as other related sub-tasks. We focus on Tasks 1a, 1b, and 2, while using the data for all tasks during training. Our model con-

siders BERTweet (Nguyen et al., 2020) for contextualized embeddings, followed by Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Capsule layers (Sabour et al., 2017) as feature extractors. We observe that adversarial learning increases performance for certain choices of hyper-parameters. Moreover, training the branch for Task 1c on the data for Task 1a improves the performance for Task 1a. In addition, an ensemble method increases the overall performance on all tasks.

The paper is organized as follows. Section 2 presents related work associated with the provided tasks. Section 3 describes our method, followed by details on the experimental setups and results. Finally, Section 5 provides conclusions and future research paths.

## 2 Related Work

Several studies (Roberts et al., 2012; Marasović and Frank, 2018; Zaharia et al., 2020) address the problem of detecting sentiments from texts by using multiple related tasks to further improve the performance of the model. Adversarial multi-task learning was introduced by Liu et al. (2017) and consists of using multiple branches for each individual task (i.e., private branches), and another branch shared among all tasks (i.e., shared branch). A discriminator trained to distinguish between features from multiple tasks is used to separate shared and task-related features in the latent representation from each branch. This is the Adversarial Shared-Private Multi-Task Learning (ASP-MTL) framework that was also employed in other works such as (Marasović and Frank, 2018), where the goal was to label opinions and the associated semantics from texts. Results showed that ASP-MTL might not achieve better results when compared to classical MTL, and there are other factors that should be taken into consideration, such as dataset split and hyper-parameters.

Other works (Zhou et al., 2019; Spiliopoulou et al., 2020) used the multi-task technique alongside adversarial learning, but in other configurations than ASP-MTL. For example, Zhou et al. (2019) employed a model that has a shared feature extractor, and then it is followed by branches for each task with an attention mechanism for the first layer from each branch. By adding new branches to the model, as well as adversarial learning, the F1-score increased when compared to the baseline

model. Spiliopoulou et al. (2020) adopted domain adaptation to improve the performance of a classifier. Their idea was to add a discriminator after the feature extractor, as a separate branch, that is used to distinguish between different domains. The experiments showed that adversarial learning aided in reducing the bias found in the dataset, thus increasing the model’s capability to generalize.

There are multiple Natural Language Processing (NLP) techniques that can be effectively employed to perform text classification in general, as well as sentiment analysis as a specific task (Paraschiv and Cercel, 2019; Tanase et al., 2020b,a; Paraschiv et al., 2020). For example, neural network methods can be used to detect propaganda in articles by pre-training models on related tasks (Vlad et al., 2019) or emotions in memes using multimodal multi-task learning (Vlad et al., 2020a,b). Other methods rely on classical machine learning methods (e.g., Support Vector Machine, Naive Bayes, or Random Forest classifiers) and can be successfully used for related tasks, such as fake news detection, obtaining accuracies over 90% on specific datasets (Dumitru and Rebedea, 2019; Busioc et al., 2020).

## 3 Method

### 3.1 Corpus

The provided dataset for SemEval-2021 Task 7 (Meaney et al., 2021) consists of 10,000 texts (6,179 texts are considered to have humor, while the rest do not present humor), written in English, with different degrees of humor, and labeled by different categories of people; therefore the labeling is highly subjective. Given the texts with humor, the scores for the Task 1b follow a Gaussian distribution, with the mean of 2.24 (out of 5) and the standard deviation of 0.56. Also, approximately 46.39% of the texts for Task 1c were labeled as controversial. Furthermore, 42.46% of the texts were rated as not being offensive at all for the last task, while the 75th percentile is 1.55 (out of 5) for the offensive samples. This means that the data is biased towards not being offensive for the Task 2.

The dataset was already split into train/dev/test sets, such that 8,000 samples were used for training, 1,000 for development, and 1,000 for testing, following similar distributions. Only the entries from training and development sets were provided with annotations during the competition. The annotations for the test set were provided after the evaluation phase ended.

### 3.2 Neural Architecture

An increasing trend of applying adversarial learning alongside MTL leads to improvements in baseline models (Liu et al., 2017; Marasović and Frank, 2018; Srivastava et al., 2020). The underlying intuition for this setting is to concurrently use potential information hidden in the correlation between multiple tasks through MTL, while storing this representation in a shared-private architecture.

Inspired by the three previously mentioned works, Figure 1 shows an overview of our proposed adversarial multi-task learning architecture, namely *AMTL-Humor*. Blue denotes the shared branch among all tasks, whereas task-specific branches are in white. The discriminative part from the network is represented in green.

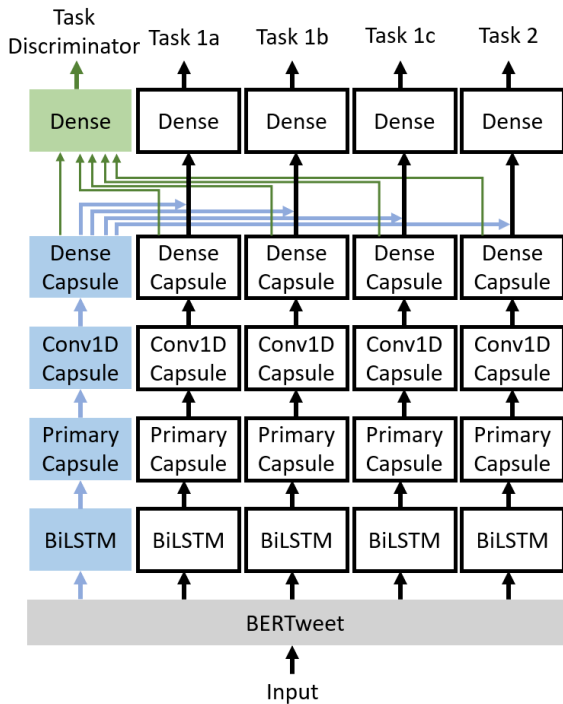


Figure 1: Overview of the AMTL-Humor architecture.

We employ a pre-trained Transformer-based model (Vaswani et al., 2017) for generating contextualized embeddings, namely BERTweet, which was trained on 850M tweets written in English. The input is preprocessed in the same way as the input on which the Transformer was trained on, by using the TweetTokenizer from NLTK toolkit (Bird et al., 2009). The representation of the input text given by the BERTweet is further fed into five branches. One branch is shared among all other tasks and learns information that is non-task-specific. The output of this branch is concatenated with each

private branch, which is then fed to a linear classifier responsible for outputting the final results. A discriminator classifier takes as input the outputs from all branches before the concatenation, having as goal to learn which representation is from what task. In this scenario, labels concerning the branch from which the discriminator took the values are known.

Each branch is composed of five layers, based on a Recurrent Neural Network (Cho et al., 2014) and Capsule layers (Lei et al., 2019; Srivastava et al., 2020). First, the *Bidirectional LSTM (BiLSTM)* layer acts as a feature extractor and encodes the high-level features. It is used to provide a latent representation of the input space that captures the dependencies from both left-to-right and right-to-left. Second, the *Primary Capsule* layer captures the local order of words in the latent space of the BiLSTM output. It encodes the hidden properties, such as positional information for words and the relations between words, from the latent representation into a vector. Third, the *Convolutional 1D Capsule* layer learns the child-parent relationships and predicts the parent Capsules in the next layer by using a dynamic routing algorithm (Sabour et al., 2017). Fourth, the *Dense (Fully Connected) Capsule* layer takes the output of the previous layer, flattens it into a list of Capsules, and then a Capsule is produced to encode the features and their probability by using routing-by-agreement. Fifth, the *Dense (Fully Connected)* layer is the classical Perceptron that takes as input the representation from the last Capsule layer and computes the final output for that branch (which can be for either classification or regression task).

The idea of using Capsule layers was first introduced in Computer Vision by Sabour et al. (2017) and was studied in various NLP tasks (Yang et al., 2018; Xiao et al., 2018; Wang, 2019; Lei et al., 2019; Srivastava et al., 2020; Zaharia et al., 2020). Capsule networks are based on how human vision works - i.e., Capsules create groups of neurons that are specialized in recognizing certain properties of the input (Sabour et al., 2017).

### 3.3 Optimization Problem

Binary cross-entropy loss is used for the classification problems (i.e., humor classification and humor controversy tasks), while the mean squared error loss function is used for regression tasks (i.e., tasks for predicting humor and offensive scores). Thus,

the loss associated with the tasks is computed as follows:

$$L_{tasks} = \sum_{i=1}^4 \alpha_i L_i \quad (1)$$

where  $\alpha_i$  represents the weight associated with the task  $i$  and  $L_i$  is the loss value for the task  $i$ .

A problem arises when employing standard MTL, namely, there is a high possibility that part of the shared information may arrive inside the task-related space and vice-versa. This problem can be alleviated by using adversarial learning to enforce the network to make the separation between the private and shared representations. Also, a generalization of the loss function (Liu et al., 2017) for training the Generative Adversarial Networks (Goodfellow et al., 2014) is used because we are dealing with multiple tasks, thus multiple classes for the discriminator:

$$L_{Adv} = \min_{\theta_S} \left( \lambda \max_{\theta_D} \left( \sum_{k=1}^5 \sum_{i=1}^{N_k} d_i^k \log[D(E(x^k))]) \right) \right) \quad (2)$$

In order to further ensure the separation between the shared and private branches, we add an orthogonality constraint function (Bousmalis et al., 2016) to the objective function during the training, which is defined as:

$$L_{Diff} = \sum_{k=1}^4 \left\| \left\| S^k \top H^k \right\|_F \right\|^2 \quad (3)$$

where  $S^k$  and  $H^k$  are the outputs of the from each branch, before the final classification layer, and  $\|\cdot\|_F$  is the Frobenius norm.

The total loss for the optimization problem is the sum of all three losses, parametrized by  $\lambda$  and  $\gamma$ , which control the importance for the adversarial and orthogonality losses, respectively:

$$L_{Total} = L_{Tasks} + \lambda L_{Adv} + \gamma L_{Diff} \quad (4)$$

### 3.4 Experimental Settings

#### 3.4.1 Text Preprocessing

The texts were not cleared; as such emojis were transformed into text to be further considered as tokens. The TweetTokenizer applies a normalization step to change noise constructs from the text (e.g., URLs) into special tokens. Missing values were replaced with 0 instead of NaN for Tasks 1b and 1c that depend on whether the text was considered to be humorous or not in Task 1a.

#### 3.4.2 Implementation Details

A public implementation of the Capsule layer<sup>1</sup> was converted from TensorFlow 1.x to TensorFlow 2.4; this updated version was also publicly<sup>2</sup>. Also, the gradient reversal layer (Ganin and Lempitsky, 2015) was used for implementing the adversarial learning between the discriminator and the shared branch, which negates the gradient for the adversarial loss during back-propagation.

The Adam optimizer (Kingma and Ba, 2014) with a learning rate of 0.001 was considered for training the networks between 10 and 20 epochs, while the batch size was set between 8 and 16. The size of the hidden state for BiLSTMs was set to 128, with a dropout rate of 0.5, as suggested by Srivastava et al. (2014). The number of filters in the Primary Capsule layer was set to 8 and the size of the kernel was set to 3x1. Also, the number of filters in the Convolutional 1D Capsule layer was set to 4 and the size of the kernel was set to 3x1. Moreover, the output size was set to 4x1 for all used Capsules, running the routing-by-agreement algorithm for 3 iterations.

During the experiments, different AMTL-Humor configurations were considered by enabling or disabling adversarial learning, the orthogonality constraints, and different parts of the architecture. Table 1 introduces our configurations, with values lower than 0.5 for hyper-parameters  $\lambda$  and  $\gamma$ , and with conditions concerning adversarial loss being prioritized over the orthogonality loss.

#### 3.4.3 Baseline Models

Two configurations were used as baseline, namely: a) a larger network (i.e., *MTL-Large*) with 512 hidden states for LSTM cells, 16 primary Capsule filters, and 8 filters for the convolutional Capsule layer with 5x1 filters, and b) a smaller model (i.e., *MTL-Small*) with the previously introduced parameters, sharing the same configurations as the other networks we have tested that use adversarial learning. The optimization step does not involve adversarial learning, nor the orthogonality constraints.

#### 3.4.4 AMTL-Humor Variants

Different models based on the ASP-MTL architecture were tested on the provided dataset. Our base model AMTL-Humor has two variants (i.e. *AMTL-*

<sup>1</sup><https://github.com/naturomics/CapsLayer>

<sup>2</sup><https://github.com/razvanalex/CapsLayer>

Model	Adversarial Training	Orthogonality Constraints	Capsule Layers	$\lambda$	$\gamma$
MTL-Large	-	-	-	0	0
MTL-Small	-	-	-	0	0
AMTL-LSTM	✓	✓	-	0.1	0.01
AMTL-Adv	✓	-	✓	0.1	0
AMTL-Humor-1	✓	✓	✓	0.1	0.01
AMTL-Humor-2	✓	✓	✓	0.5	0.1
AMTL-T1a-Twice*	✓	✓	✓	0.05	0.01

Table 1: The settings for each configuration. The star (\*) indicates that the model uses the third branch (i.e., for Task 1c) to solve Task 1a instead. Note that two branches are used for solving Task 1a.

*Humor-1* and *AMTL-Humor-2*), as presented in Table 1. The only difference between these is the choice for the  $\lambda$  and  $\gamma$  parameters. Also, *AMTL-Adv* is a model without orthogonality constraints, whereas *AMTL-LSTM* does not contain the Capsule layers in the architecture. Throughout our experiments, we observed that the network always outputs 0 on Task 1c for all inputs; therefore, the model does not manage to learn anything. As such, the *AMTL-T1a-Twice* configuration uses the Task 1a branch twice for both the first and third branches.

### 3.4.5 Ensembles

Ensemble learning was also considered; the modal value (i.e., the most frequent class) is taken for classification tasks, while the average of scores is used for regression tasks. *Ensemble-1* combines the results obtained by *MTL-Large*, *AMTL-Humor-1*, *AMTL-T1a-Twice*, and *AMTL-Adv* respectively, while *Ensemble-2* adds to the previous list the models *MTL-Small*, *AMTL-LSTM*, and *AMTL-Humor-2*.

### 3.4.6 Evaluation Metrics

F1-score and accuracy were used to evaluate the classification tasks, whereas root mean squared error (RMSE) was used to assess performance on the regression tasks from the SemEval-2021 Task 7 competition.

## 4 Results

Table 2 presents the results obtained on both development and test sets. On the development set, we observe that increasing the values of the parameters impacts more the regression tasks (i.e., higher RMSE values). Using only adversarial learning, without orthogonality constraints seems to add a small improvement in our case. Removing the ad-

versarial learning has a very small negative impact on Task 1a, and a decrease of RMSE for the regression tasks. Removing only the Capsule layers seems to generate an improvement for this development set. The *AMTL-T1a-Twice* configuration achieves the highest F1-score due to the symmetry of our architecture. The increase in performance is notable, more than 2% when compared to *AMTL-Humor-1*.

We observe that our baseline models achieve on the test set similar scores for Task 1a, whereas differences exist for the other two tasks, with no clear better configuration. The larger network tends to learn better on Task 1b, whereas RMSE is worse on Task 2 when compared to *MTL-Small*. All models that use our adversarial multi-task framework show a small improvement over the baselines on Task 2; nevertheless, almost all models perform worse on Task 1b, especially our system (i.e., *AMTL-Humor-1*) that registers the highest RMSE scores. All adversarial models achieve or the Task 1a similar results, or even better when compared to the baseline models, and this was our main focus during the training.

The *AMTL-T1a-Twice* model manages to obtain the highest F1-score on one of the two branches for Task 1a. However, accuracy and F1 scores were low when making predictions for Task 1.c (46.34% and 61.18%, respectively). These low performance values may be indicative of bias in the dataset towards either 0 or missing values for this task.

Ensemble models complement the inner configurations and provide more stable predictions. *Ensemble-1* was the model submitted during the competition for evaluation. The differences between our *Ensemble-1* configuration and the best models reported for each task were the following:



Model	Development Set				Test Set			
	Task 1a		Task 1b	Task 2	Task 1a		Task 1b	Task 2
	F1 (%)	Acc. (%)	RMSE	RMSE	F1 (%)	Acc. (%)	RMSE	RMSE
MTL-Large*	94.80	<b>93.40</b>	0.6434	0.7390	95.05	93.90	0.6359	0.5891
MTL-Small	93.08	90.90	0.6747	0.6649	95.07	93.80	0.6699	0.5796
AMTL-LSTM	94.12	92.50	0.6939	0.6738	95.39	94.30	0.6569	0.5552
AMTL-Adv*	93.52	92.10	0.6979	0.7053	93.61	92.40	0.6882	0.5631
AMTL-Humor-1*	93.27	91.40	0.7294	0.7294	94.93	93.80	0.7116	0.5616
AMTL-Humor-2	93.75	92.10	0.7426	0.6759	95.32	94.30	0.7151	0.5680
AMTL-T1a-Twice	<b>95.88</b>	92.09	0.6977	0.6881	<b>96.29</b>	92.85	0.6774	0.5772
Ensemble-1*	94.05	92.50	<b>0.6361</b>	0.6656	95.66	94.70	0.6200	0.5318
Ensemble-2	94.54	93.10	0.6383	<b>0.6497</b>	95.82	<b>94.90</b>	<b>0.6164</b>	<b>0.5270</b>

Table 2: The results obtained on both development set (left) and test set (right). The best scores are marked in bold. The star (\*) indicates the models we used for the submissions during the evaluation phase.

2.88% less F1-score and 3.5% less accuracy for Task 1a, 0.124 higher RMSE for Task 1b, and 0.119 higher RMSE for Task 2. During the post-evaluation phase, we assessed the Ensemble-2 configuration that performs better than Ensemble-1 on all tasks by a small margin (i.e., less than 1%).

#### 4.1 Visualizations

t-SNE visualizations (Van der Maaten and Hinton, 2008) were considered to better grasp the adequacy of our model. t-SNE is an algorithm used to visualize high-dimensional data into a two-dimensional representation by minimizing the Kullback-Leibler divergence between the joint probability distributions for both lower-dimensional and higher-dimensional data from the input. Since the output from BERTweet has 59,904 dimensions (i.e., 78 tokens in a sequence, each having 768 features given by the hidden units), we first reduced the representation to 100 dimension by applying a Principal Component Analysis (Jolliffe and Cadima, 2016). This way, t-SNE runs faster, without losing too much information when considering the principal components.

The results using the Ensemble-2 configuration are shown in Figure 2 for Tasks 1a, 1b, and 2. Each point represents the embedding of the input sample, while the colors represent the classes or the values for each sample. Based on the ground-truth visualizations (left side), we can observe that the points are not linearly separable. Our best configuration was capable to learn the inner representation of the input data. For the offensive rating task, the examples with higher scores are in the middle of the manifold. We can also observe for the offensive

rating task that the majority of classes are biased towards lower rather than higher values.

## 5 Conclusions and Future Work

In this work, we present AMTL-Humor, an adversarial multi-task learning method to deal with the problem of detecting and rating humor and offense for SemEval-2021 Task 7. More specifically, our model is inspired by the ASP-MTL framework and considers BiLSTM and Capsule layers as feature extractors on top of the BERT layer that provides contextualized embeddings.

Two ensemble models were created using variations of our AMTL-Humor architecture. These configurations were trained either on different settings (such as adversarial learning) or by modifying the structure of the branches. We observed that adversarial learning might perform better than other architectures with similar structures, while considering specific configurations and tasks. Another improvement was observed when the model was trained on two tasks using the same branch, namely the AMTL-T1a-Twice model. This approach of duplicating branches may be used in other context as it adds redundancy to the network and supports generalization. Finally, the best results were obtained by using an ensemble over all trained models.

In the future, we aim to study how performance can be improved, especially on Task 1c, by adapting the hierarchical multi-task learning technique (Søgaard and Goldberg, 2016) to the AMTL-Humor architecture.

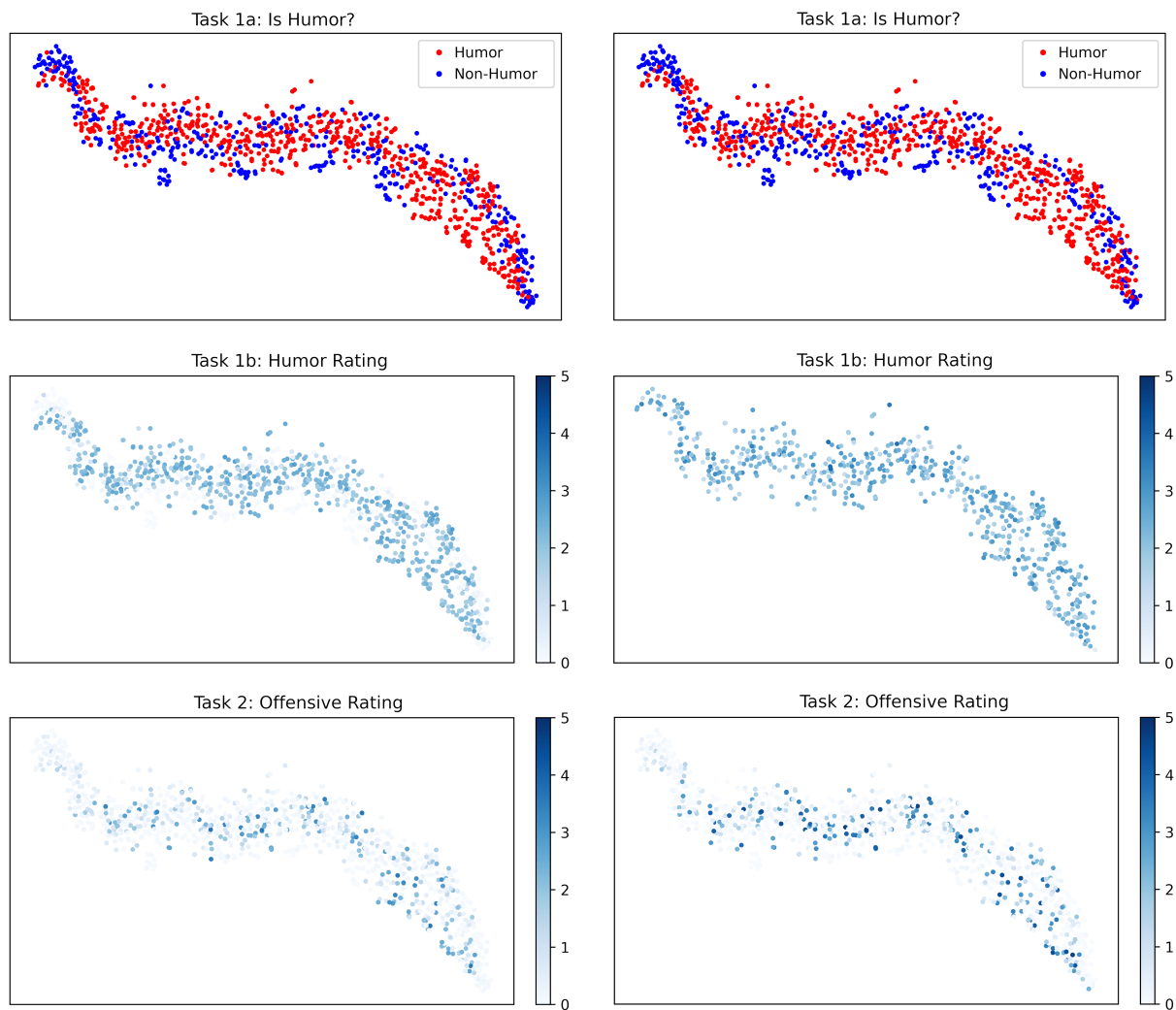


Figure 2: t-SNE projection for the development set data on the output of the BERTweet layer. On left are the plots for predicted outputs, whereas on right are the corresponding plots for ground-truth. Best viewed in color.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O’Reilly Media.
- Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain separation networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 343–351.
- Costin Busioc, Stefan Ruseti, and Mihai Dascalu. 2020. A literature review of nlp approaches to fake news detection and their applicability to romanian-language news analysis. *Revista Transilvania*, (10).
- Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.
- Vlad Cristian Dumitru and Traian Rebedea. 2019. Fake and hyper-partisan news identification. In *RoCHI*, pages 60–67.
- Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised domain adaptation by backpropagation. In *International conference on machine learning*, pages 1180–1189. PMLR.
- Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2672–2680.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

- Ian T Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kai Lei, Qiurai Fu, and Yuzhi Liang. 2019. Multi-task learning with capsule networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.
- Pengfei Liu, Xipeng Qiu, and Xuan-Jing Huang. 2017. Adversarial multi-task learning for text classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- Ana Marasović and Anette Frank. 2018. Srl4orl: Improving opinion role labeling using multi-task learning with semantic role labeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 583–594.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14.
- Andrei Paraschiv and Dumitru-Clementin Cercel. 2019. Upb at germeval-2019 task 2: Bert-based offensive language classification of german tweets. In *KONVENS*.
- Andrei Paraschiv, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020. Upb at semeval-2020 task 11: Propaganda detection with domain-specific trained bert. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1853–1857.
- Kirk Roberts, Michael A Roach, Joseph Johnson, Josh Guthrie, and Sanda Harabagiu. 2012. Empatweet: Annotating and detecting emotions on twitter. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 3806–3813.
- Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3859–3869.
- Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235.
- Evangelia Spiliopoulou, Eduard Hovy, Alexander G Hauptmann, et al. 2020. Event-related bias removal for real-time disaster events. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3858–3868.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- Saurabh Srivastava, Puneet Agarwal, Gautam Shroff, Lovekesh Vig, and Vidya Vikas. 2020. Capsule based neural network architecture to perform completeness check for patent eligibility process. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.
- Mircea-Adrian Tanase, Dumitru-Clementin Cercel, and Costin Chiru. 2020a. Upb at semeval-2020 task 12: Multilingual offensive language detection on social media by fine-tuning a variety of bert-based models. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2222–2231.
- Mircea-Adrian Tanase, George-Eduard Zaharia, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020b. Detecting aggressiveness in mexican spanish social media content by fine-tuning transformer-based models. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2020) co-located with 36th Conference of the Spanish Society for Natural Language Processing (SEPLN) 2020*, pages 236–245.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010.
- George-Alexandru Vlad, Mircea-Adrian Tanase, Cristian Onose, and Dumitru-Clementin Cercel. 2019. Sentence-level propaganda detection in news articles with transfer learning and bert-bilstm-capsule model. In *Proceedings of the Second Workshop on Natural Language Processing for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 148–154.

- George-Alexandru Vlad, George-Eduard Zaharia, Dumitru-Clementin Cercel, Costin Chiru, and Stefan Trausan-Matu. 2020a. Upb at semeval-2020 task 8: Joint textual and visual modeling in a multi-task learning architecture for memotion analysis. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1208–1214.
- George-Alexandru Vlad, George-Eduard Zaharia, Dumitru-Clementin Cercel, and Mihai Dascalu. 2020b. Upb@ dankmemes: Italian memes analysis-employing visual models and graph convolutional networks for meme identification and hate speech detection. In *Proceedings of Seventh Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2020), Online. CEUR.org*.
- Mingxuan Wang. 2019. Towards linear time neural machine translation with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 803–812.
- Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. 2018. Mcapsnet: Capsule network for text with multi-task learning. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 4565–4574.
- Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating capsule networks with dynamic routing for text classification. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3110–3119.
- George-Eduard Zaharia, George-Alexandru Vlad, Dumitru-Clementin Cercel, Traian Rebedea, and Costin Chiru. 2020. Upb at semeval-2020 task 9: Identifying sentiment in code-mixed social media texts using transformers and multi-task learning. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1322–1330.
- Xiabing Zhou, Zhongqing Wang, Shoushan Li, Guodong Zhou, and Min Zhang. 2019. Emotion detection with neural personal discrimination. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5502–5510.

# Team\_KGP at SemEval-2021 Task 7: A Deep Neural System to Detect Humor and Offense with Their Ratings in the Text Data

Anik Mondal

Indian Institute of Technology Kharagpur  
pjanik2000@iitkgp.ac.in

Raksha Sharma

Indian Institute of Technology Roorkee  
raksha.sharma@cs.iitr.ac.in

## Abstract

This paper describes the system submitted to SemEval-2021 Task-7 for all four subtasks. Two subtasks focus on detecting humor and offense from the text (binary classification). On the other hand, the other two subtasks predict humor and offense ratings of the text (linear regression). In this paper, we present two different types of fine-tuning methods by using *linear layers* and *bi-LSTM* layers on top of the pre-trained BERT model. Results show that our system is able to outperform baseline models by a significant margin. We report F1 scores of 0.90 for the first subtask and 0.53 for the third subtask, while we report an RMSE of 0.57 and 0.58 for the second and fourth subtasks, respectively.

## 1 Introduction

Automatic humor and offense detection have considerable importance in the modern age, especially in chatbots and virtual assistants (Augello et al., 2008). Detection of humor and offense in the text is a highly subjective phenomenon that varies with age, gender, race, socio-economic status, etc. Therefore, it is one of the most challenging research fields in Natural Language Processing (Taylor, 2009).

Task-7 of SemEval 2021 (Meaney et al., 2021) is concerned with humor detection (binary classification) primarily. The first subtask is to check if a text is humorous, another subtask follows up: how humorous is the text (rated from 0-5). We also have to predict whether the text is generally offensive (binary classification task). The last subtask predicts how generally offensive a text is (rated from 0-5), regardless of whether it is classed as humorous or offensive overall. The dataset consists entirely of English texts.

The traditional methods deployed earlier for humor detection include Support Vector Machine

(SVM) with RBF kernel, Random Forest Classifier, and SGD with Logical Classifier, all of which provide modest results (de Oliveira and Rodrigo, 2015). Recently more state-of-the-art transformers have provided better results (Weller and Seppi, 2019). The system presented in this paper is fine-tuning one of the best and most popular state-of-the-art models, Bidirectional Encoder Representations from Transformers (BERT). We have used pre-trained BERT embeddings to represent the words and used the features of the last layer of the 12-layers BERT Model to *detect* and *rate* both *humor* and *offense*. BERT can model complex interactions between different levels of hierarchical information (Tenney et al., 2019). This task is perhaps the first task that focuses on the humor and offense ratings and combines humor and offense detection.

Our model obtained an F1 score of 0.92 and 0.56 for the first and the third subtasks, respectively, on the test data. We obtained an RMSE of 0.57 and 0.58 for the second and fourth subtasks, respectively, on the test data.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 describes the dataset for the task. Section 4 provides the experimental setup and evaluation metric. Section 5 presents the systems implemented to address the task. Section 6 shows the results and Section 7 concludes the paper.

## 2 Related Work

We found the following works very relevant for the task to be addressed, *i.e.*, humor and offense detection.

**Lexical Syntactic Feature (LSF):** This architecture (Chen et al., 2012) was used to detect offensive language in social media. It bridges the gap between message-level and user-level offensive language detection. In particular, this paper

incorporates a user’s writing style, structure, and specific cyberbullying content as features to predict the user’s potentiality to send out offensive content.

**Netflix-style collaborative filtering:** This method (Gultchin et al., 2019) identifies a mean humor direction, analyses sense-of-humor word embeddings to predict individual differences in word humor. It proposes Netflix-style collaborative filtering to predict humor ratings.

**BERT-base and BERT-large:** BERT models have been used previously for automatic humor detection and scoring (Mao and Liu, 2019). The pre-trained model is fine-tuned on the available data, producing appreciable results.

Our system lies in close proximity to the last work. We have fine-tuned a pre-trained BERT model as well. The difference lies in the fine-tuning process. Different approaches have been used to building the neural network, varying from linear layer approach to bi-LSTM approach, for both the classification and regression tasks.

### 3 Dataset

The dataset used in this paper is obtained from Task 7 of SemEval 2021. In this task, the organizers collected labels and ratings from a balanced set of age groups from 18 to 70. The annotators also represented a variety of genders, political stances, and income levels. The training set consists of 8000 texts, and the development and test datasets consist of 1000 texts each. The variation of the number of texts vs. word length of texts is shown in Figure 1.

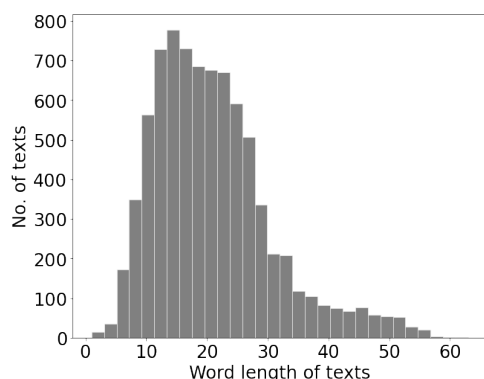


Figure 1: The distribution of lengths of the texts from the training set.

Each text is represented by a unique ID, and each subtask has a separate column for each text. The annotators were asked:

1. Is the intention of this text to be humorous?
2. (If it is intended to be humorous according to the rater) How humorous do you find it? (1-5)

The annotators could also give a rating of 0 to the second question if they do not get it due to the text’s structure or content. The label of the first task is *is\_humor* and is based on the majority class given by 20 annotators. In case of a tie, the humor label was selected. The humor rating is based on the annotators’ average rating, under the label *humor\_rating*.

Table 1 gives an insight into the distribution of *is\_humor* label. The distribution of *humor\_rating* is shown in Figure 2.

<i>is_humor</i>	No. of texts
0	4932
1	3068

Table 1: The distribution of texts considered either humorous (1) or not (0).

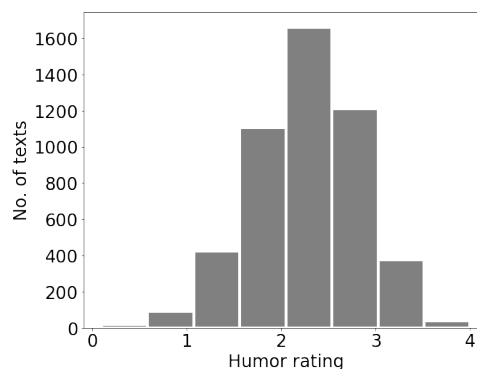


Figure 2: The distribution of *humor\_rating* of the texts from the training set.

The annotators were further asked:

3. Is this text generally offensive? (0 or 1)
4. (If the rater considers the text to be generally offensive) How generally offensive is the text? (1-5)

By generally offensive, the organizers mean that the text targets a person or group for merely belonging to a specific group and ask users if they think that a significant number of people would find this offensive. If the variance of a text was higher than the median variance of all texts, the humor of the text was labeled as controversial under the label *humor\_controversy*. It is a binary classification task.

In the last question, we consider the ratings 1-5 and also consider a no rating to be 0. This score (average of all the ratings) was calculated regardless of whether the text is classed as humorous or offensive overall, under the label *offense\_rating*.

Table 2 gives an insight into the distribution of *humor\_controversy* label. The distribution of *offense\_rating* is shown in Figure 3.

<i>humor_controversy</i>	No. of texts
0	2467
1	2465

Table 2: The distribution of texts considered either offensive (1) or not (0).

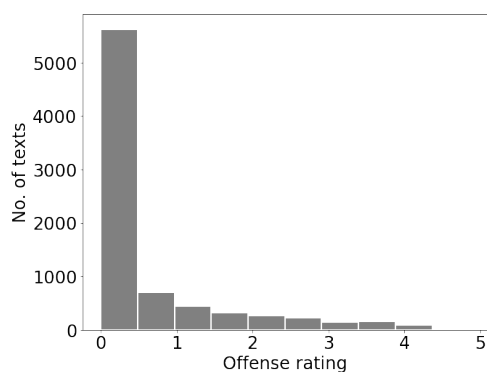


Figure 3: The distribution of *offense\_rating* of the texts from the training set.

## 4 Experimental Setup and Evaluation Metric

**Preprocessing:** A data pipeline preprocesses the raw data to remove irrelevant features. The stop-words and emojis (if any) are removed from the data, and the output of the pipeline is lower-case stemmed word sequences. The stop-words are selected from the NLTK stop-words list (Sarica and Luo, 2020).

**Max Sequence Length:** From Figure 1, we observe that there is no sequence having a length greater than 70 words. So, we restrict our *max\_seq\_length* parameter of the BERT Tokenizer to 70. We pad the sentences to the maximum sequence length.

**Adam Optimizer:** For optimizing the model, we use Adam Optimizer. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems and

works well on deep neural networks (Zhang, 2018). For subtasks-1 and 3, we used a learning rate of  $2e-5$ , and for subtasks-2 and 4, a learning rate of  $1e-5$  was used.

**Experimental Tools:** We used Google Colab to run the experiments, which provided us GPU to run our model. The external libraries used are publicly available Transformers (version 3.0.0) from the PyTorch HuggingFace API<sup>1</sup> and Python-based Scikit-learn package.

**Evaluation Metric** The metric for the classification tasks (subtask 1 and 3) is F1 score.

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (1)$$

For the regression tasks (subtask 2 and 4), the metric is Root Mean Square Error (RMSE).

$$RMSE = \sqrt{\sum_{i=1}^N \frac{(Predicted_i - Actual_i)^2}{N}} \quad (2)$$

## 5 System

The task organizers suggested the following baseline models:

1. Classification task: For the classification tasks, the baseline strategy is a Naive Bayes model with bag of words features.
2. Regression task: The baseline strategy proposed for the regressions tasks is the Support Vector Regression (SVR) model.

BERT has proven promising for many NLP tasks (Devlin et al., 2019). Our system implements fine-tuning strategies on pre-trained BERT architecture. It is a bidirectional transformer pre-trained using a combination of masked language modeling objective and next sentence prediction on a large corpus comprising the Toronto Book Corpus and Wikipedia.

### 5.1 BERT base model

We have experimented with the BERT-base model from PyTorch HuggingFace API. It is the bare BERT Model (BertModel) transformer outputting raw hidden-states without any specific head on top. The 768 hidden features are extracted from the last layer of the 12-layered BertModel.

<sup>1</sup><https://huggingface.co/>.

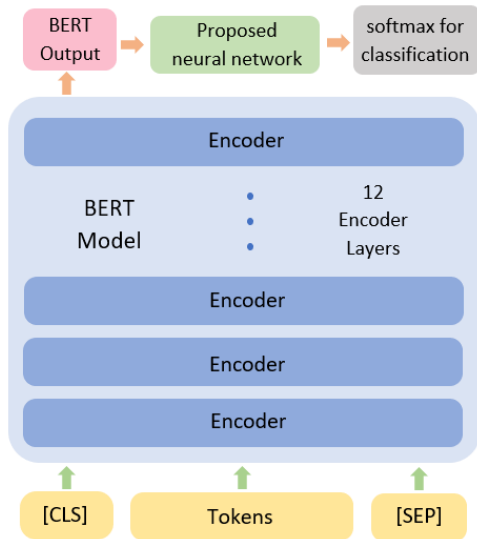


Figure 4: The proposed model architecture for the classification subtasks

### 5.1.1 Classification Tasks

We have experimented with two independent approaches.

- **Linear approach:** After experimenting with single and double layers, we decided to use three linear layers on top of the features, as it produced better results. The layers are of the dimensions  $768 \times 512$ ,  $512 \times 256$ , and  $256 \times 2$  respectively. Rectified linear unit (ReLU) is used as the nonlinear activation function at the first two layers' output, followed by a dropout regularization of 0.1.
- **Bi-LSTM approach:** We use one bi-LSTM (Bi-directional long short term memory) layer of 512 dimensions at first instead of the initial linear layer. We keep the rest of the architecture the same (including the activation and regularization), altering the dimensions as required.

The output layer is a softmax layer for the classification job (for both approaches). The reason behind experimenting with the Bi-LSTM model is that it fully considers the context information and can better obtain the text representation of the comments (Xu et al., 2019).

In the BERT training process, the model receives pairs of sentences as input in a specific format. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence, with the words in the middle being converted to tokens, as shown in Figure 4.

### 5.1.2 Regression Tasks

Three neural network architectures have been tried on top of the BERT features enumerated below for the regression tasks.

- **3 linear layers:** The same linear layer architecture used for the classification tasks is implemented here; only the dimensions of the last layer are altered appropriately. 3 linear layers of dimensions  $768 \times 512$ ,  $512 \times 256$ , and  $256 \times 1$  respectively are used. ReLU is applied as the activation function at the output of the first two layers, followed by a dropout regularization of 0.1. The loss function used for this regression task is the mean squared error (MSE) loss function.
- **One linear layer:** Only one linear layer is used in this approach of the dimension  $768 \times 1$ . MSE loss function is used for the cross-entropy loss.
- **Bi-LSTM layers:** One bi-LSTM layer of 512 dimensions is applied on top of the extracted features from BertModel. On top of that, a linear layer is used to predict the ratings. The same loss function as used in the first two approaches is applied here.

## 6 Results and Analysis

For each subtask, we trained our data on the entire training set of 8000 texts. We used the development set of 1000 texts as cross-validation data. The results tabulated in this section are reported on the gold test set of 1000 texts.

Each proposed model for all the subtasks was run for 10 epochs with a batch size of 32. Hyper-parameters are discussed in Section 5.

Method	Accuracy	F1 score
Linear approach	0.9030	0.9233
bi-LSTM approach	0.8970	0.9177
Naive Bayes	0.8570	0.8840

Table 3: Accuracy and F1 score for the models trained on the humor classification task

The results of the considered BertModel and baseline methods for the *is\_humor* subtask-1 are summarized in Table 3 in terms of the F1 score and accuracy. Table 4 shows the results of the *humor\_controversy* subtask-3. We observe that the linear approach works better than the bi-LSTM



Method	Accuracy	F1 score
Linear approach	0.5301	0.5628
bi-LSTM approach	0.5203	0.5455
Naive Bayes	0.4374	0.4624

Table 4: Accuracy and F1 score for the models trained on the humor controversy task

approach in both cases. On the other hand, both the proposed models perform better than the baseline model (Naive Bayes with bag of words features).

Method	RMSE
3 linear layers approach	0.5741
Single linear layer approach	0.5847
Bi-LSTM layer approach	0.5694
SVR	0.8609

Table 5: RMSE for the models trained on the humor rating task

Method	RMSE
3 linear layers approach	0.5936
Single linear layer approach	0.6082
Bi-LSTM layer approach	0.5800
SVR	0.6415

Table 6: RMSE for the models trained on the offense rating task

Tables 5 and 6 represent the results of all the three approaches along with the given baseline (SVR) for the regression subtasks in terms of RMSE. Results show that the 3-layered method works better than the single-layer method. We observe that among all three methods, the bi-LSTM approach works the best for both the subtasks. Our proposed methods have significantly lower RMSEs than the baseline SVR model.

In the humor detection subtask, we ranked 48th with the leaders achieving an F1 score of 0.98. We achieved a rank of 23 in the humor rating subtask, leaders getting an RMSE of 0.49. In the humor controversy subtask, the leaders got an F1 score of 0.63, giving us a rank of 15. In the last subtask of offense rating, we achieved a rank of 38, with the leaders getting an RMSE of 0.41.

The section where our model struggles the most is detecting underlying sarcasm in sentences, especially where the context is explored for the first time. For example, in the text: “I asked my

North Korean friend how it was there... he said he couldn’t complain.”, our model classifies the text as not humorous however in the gold set the text has been classified as humorous. This is one of several examples where our classifier mislabeled the text. The same problem exists with other sub-tasks too.

## 7 Conclusion

In this paper, we describe a system developed to address the SemEval Task-7. The task has four subtasks, *viz.*, detecting humor, detecting offense, predicting humor rating, and predicting offense rating. Our system is able to perform all four subtasks with varying levels of performance for each task. Our system deploys linear layers and bi-LSTM layers independently to process the features produced by the BERT model. Results show that our system using BERT with Linear layers outperforms the baseline model by a significant margin for the first and the third subtasks. On the other hand, the bi-LSTM layers-based system gives the best performance for the other two fine-grained rating prediction tasks.

## References

- A. Augello, G. Saccone, S. Gaglio, and G. Pilato. 2008. [Humorist bot: Bringing computational humour in a chat-bot system](#). In *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, pages 703–708.
- Y. Chen, Y. Zhou, S. Zhu, and H. Xu. 2012. [Detecting offensive language in social media to protect adolescent online safety](#). In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, pages 71–80.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Limor Gultchin, Genevieve Patterson, Nancy Baym, Nathaniel Swinger, and Adam Kalai. 2019. [Humor in word embeddings: Cockamamie gobbledegook for nincompoops](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2474–2483. PMLR.
- Jihang Mao and W. Liu. 2019. [A bert-based approach for automatic humor detection and scoring](#). In *IberLEF@SEPLN*.

- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and of-fense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Luke de Oliveira and A. Rodrigo. 2015. Humor detec-tion in yelp reviews.
- Serhad Sarica and Jianxi Luo. 2020. [Stopwords in tech-nical language processing](#).
- J. M. Taylor. 2009. [Computational detection of humor: A dream or a nightmare? the ontological semantics approach](#). In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 429–432.
- Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. [Bert rediscovers the classical nlp pipeline](#).
- Orion Weller and Kevin Seppi. 2019. [Humor detection: A transformer gets the last laugh](#).
- G. Xu, Y. Meng, X. Qiu, Z. Yu, and X. Wu. 2019. [Sen-timent analysis of comment texts based on bilstm](#). *IEEE Access*, 7:51522–51532.
- Z. Zhang. 2018. [Improved adam optimizer for deep neural networks](#). In *2018 IEEE/ACM 26th Interna-tional Symposium on Quality of Service (IWQoS)*, pages 1–2.

# ZYJ at SemEval-2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense with ALBERT-Based Model

Yingjia Zhao

Yunnan University / Yunnan, P.R. China  
zyj1309700118@gmail.com

Xin Tao

Yunnan University / Yunnan, P.R. China  
taoxinwy@126.com

## Abstract

Although humorous language can bring joy to people, it is also easy to cause offense. Therefore, in order to effectively identify whether a sentence is humorous or offensive, the system needs to obtain abundant semantic information. This article introduces the submission of subtask 1 and subtask 2 that we participate in SemEval-2021 Task 7: HaHackathon: Detecting and Rating Humor and Offense, we use a model based on ALBERT that uses ALBERT as the module for extracting text features. We modify the upper layer structure by adding specific networks to better summarize the semantic information. Finally, our system achieves an F-Score of 0.9348 in subtask 1a, RMSE of 0.7214 in subtask 1b, F-Score of 0.4603 in subtask 1c, and RMSE of 0.5204 in subtask 2.

## 1 Introduction

A sense of humor is a positive psychological quality that can make people feel better about themselves, relieve stress, and gain a sense of connection with others. Humor can help bring about happiness experiences and optimism. Humor, by its nature, stirs up emotions. The speaker presents the audience with an unexpected conflict. The audience feels nervous and anticipatory, and at the same time feels pleased and released. Humor is also a highly subjective phenomenon, with age, gender, and socioeconomic status is known to influence the perception of jokes. That's why some things make people laugh and others don't. When the brain lacks the cognitive resources to accurately understand the context in which a joke takes place, it generalizes it into an everyday behavior, and the benign offense becomes hostile aggression. Like most metaphorical languages, humor emphasizes multiple word meanings, cultural knowledge, and pragmatic capabilities, making it a challenging task to detect humor and offense in a sentence.

SemEval-2021 Shared Task 7: HaHackathon: Detecting and Rating Humor and Offense (Meaney et al., 2021) shared task has two subtasks. Subtask 1 emulates previous humor detection tasks in which all ratings were averaged to provide mean classification and rating scores. Subtask 1a: predict if the text would be considered humorous (for an average user). This is a binary task. Subtask 1b: if the text is classed as humorous, predict how humorous it is (for an average user). The values vary between 0 and 5. Subtask 1c: if the text is classed as humorous, predict if the humor rating would be considered controversial, i.e. the variance of the rating between annotators is higher than the median. This is a binary task. Subtask 2 aims to predict how offensive a text would be (for an average user) with values between 0 and 5. This score was calculated regardless of whether the text is classed as humorous or offensive overall.

In this paper, we use ALBERT: A Lite BERT for self-supervised Learning of Language (Lan et al., 2019) as the module for extracting sentence semantic information. The features extracted by ALBERT are then further processed through a specific structure. Besides, for subtasks 1a and 1c, since it is a binary classification problem, we use k-fold stratified sampling to reinforce the training process. The rest of the paper is organized as follows. Part 2 gives a brief introduction to the relevant work. Part 3 describes the dataset and our approach. Part 4 describes the hyperparameters of the study method used and our results. Finally, the fifth part summarizes our work.

## 2 Related Work

Computational research in the field of humor detection has been going on for some time, and the diversity of tasks allows for different analyses according to the type and expression of humor. In

previous shared tasks, T3 team (Vanroy et al., 2020) used the pre-trained language model Roberta in the SemEval-2020 shared task7 on Assessing the Fun-  
 niness of Edited News Headlines (Hossain et al., 2020) to learn the latent features in news headlines and predict how funny each headline is. UniTue-  
 bingenCL team (Ammer and Grüner, 2020) used a Ridge Regression model using Elmo and Glove embeddings as well as Truncated Singular Value  
 Decomposition at SemEval-2020 Task 7: Humor  
 Detection in News Headlines. Humor and emotions  
 such as offense and hatred are not only different but  
 also related to one another. It is often challenging  
 to classify them accurately. Badlani et al. (2019)  
 proposed a composite two-step model. In the first  
 step, features related to irony, humor, hate speech,  
 and emotion are extracted, and in the second step,  
 these features are combined to classify emotions.  
 This multi-step method is better than a single step.  
 Models that predict sentiment have better empirical  
 performance in sentiment classification.

Sentiment analysis of humor data requires a  
 deep semantic understanding of the text, and sig-  
 nals of nuances in the language may enhance or  
 completely change the sentiment of the sentence.  
 Morales and Zhai (2017) proposed a generative lan-  
 guage model based on incongruity theory to model  
 humorous text, using background text sources, such  
 as Wikipedia entry descriptions, and being able to  
 construct multiple features to identify humorous  
 comments. Besides, Deep Learning (DL) (Good-  
 fellow et al., 2016) methods of multi-layer Neural  
 Networks (NN) (Mikolov et al., 2011) stacked was  
 also a common method. Ortega-Bueno et al. (2018)  
 used a recurrent neural network(RNN) (Mikolov  
 et al., 2010) that combines language features and  
 attention-based to classify Spanish tweets as hu-  
 morous or not and predict how funny they are. The  
 attention (Vaswani et al., 2017) layer helps calcu-  
 late the contribution of each term to the target hu-  
 mor category. In recent years, various pre-training  
 models based on Transformer have shown outstand-  
 ing performance, and some researchers have also  
 applied them in the field of humor detection. Weller  
 and Seppi (2019) used a Transformer framework  
 to evaluate whether a joke was humorous, and per-  
 formed well on the short joke and pun datasets.

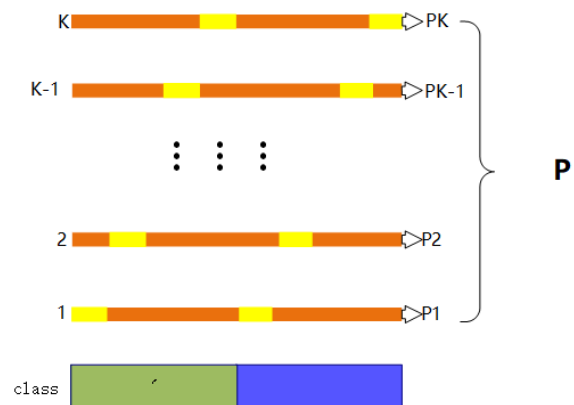


Figure 1: K-fold stratified sampling to the training set

### 3 Methodology and Data

#### 3.1 Data description

The organizer of Semeval-2021 Shared Task 7 provided a complete training data set for our system, including 8000 pieces of data for the training set, 1000 for the validation set, and 1000 for the test set. Each piece of data has four tags.

In our experiment, for subtasks 1a and 1c, we use a Stratified-K-fold technique to randomly segment all combined training datasets. As shown in Figure 1, we use the Stratified-K-fold cross-validation instead of the ordinary K-fold cross-validation. Stratified-K-fold can ensure that the proportion of each class in the generated training set and validation set is consistent with the original training set, thus avoiding the generated data distribution disorder. In this experiment, we set the value of K as 5.

#### 3.2 Description of the system

Our model is one based on ALBERT, which is shown in Figure 2. BERT (Devlin et al., 2018) has good performance, but too many parameters and long training time are also its disadvantages. ALBERT is designed by Google mainly to solve the problem of BERT, which is a simplified model based on BERT. ALBERT solves these problems by using two parameter reduction techniques, one of them is cross-layer parameter sharing, in order to avoid quantity increases along with the network depth; factorized embedding parameterization is another approach, by putting a big word embedded matrix is decomposed into two small matrix makes the relationship between the size of the hidden layer and dictionary apart, thus, when the size of the hidden layer is increased, the parameter size of

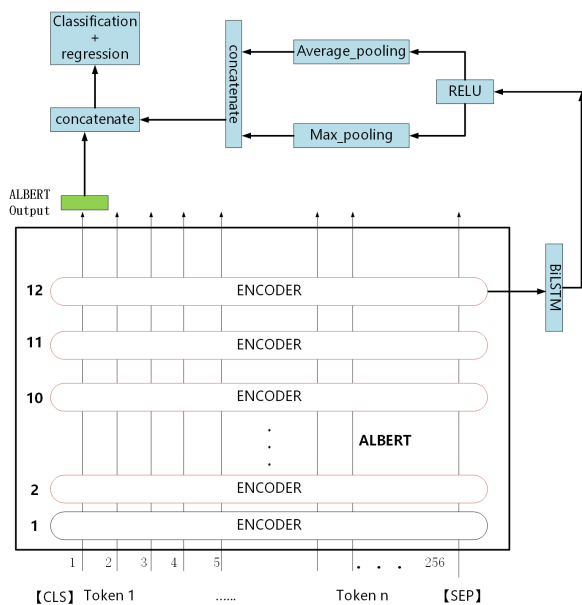


Figure 2: Schematic overview of the architecture of our model

the vocabulary embedding will not be significantly increased.

We input the preprocessed text into the model through the input layer, and then carry out vector representation. Position vector representation, text vector representation, and word vector representation make up the vector representation. Then the ALBERT model takes the sum of the input embedding, position encoding, and token type embedding as input, which is further processed by the Transformer Encoder module. After that, we input the output of the last ALBERT hidden layers into BiLSTM. Then, through the Relu function, the vector is nonlinearly mapped to the lower dimension. After average pooling and maximum pooling are achieved to obtain the feature vector, we concatenate the average pooling and maximum pooling output. Finally, the feature vector will concatenate with the original output of ALBERT, the classification or regression task is then performed.

## 4 Experiment and results

### 4.1 Experiment setting

In this experiment, `albert_base_v2` is used. After adding a new module based on ALBERT, the whole model is fine-tuned. The main hyper-parameters we adjust are the maximum sequence length, the learning rate, the gradient accumulation steps, and batch size. As is shown in Table 1.

<b>maximum sequence length</b>	<b>learning rate</b>
128	2e-5
<b>gradient accumulation</b>	<b>steps batch size</b>
4	4

Table 1: Details of the hyper-parameters.

<b>Team Name: ZYJ</b>	
<b>Task 1a Humor Detection</b>	
F-Score	Rank
0.9348	41
<b>Task 1b Average Humor Score</b>	
RMSE	Rank
0.7214	43
<b>Task 1c Humor Controversy</b>	
F-Score	Rank
0.4603	33
<b>Task 2 Average Offensiveness Score</b>	
RMSE	Rank
0.5204	31

Table 2: The results of our methods.

## 4.2 Results

According to the leaderboard provided by the organizer, our team's F-score is 0.9348, ranking 41st place in subtask 1a Humor Detection. RMSE is 0.7214, ranking 43rd place in subtask 1b Average Humor Score. F-Score is 0.4603, ranking 33rd place in subtask 1c Humor Controversy. RMSE is 0.5204, ranking 31st place in subtask 2 Average Offensiveness Score. As shown in Table 2.

## 5 Conclusion

In this task, we detect and rate humor and offense using a deep-learning-based model. In the construction of the model, we use ALBERT as a module of the model and add a custom network structure to further process the extracted feature vector. As for the classification task, we perform Stratified-K-Fold cross-validation based on the model and get the optimal value through the voting mechanism. Although our system has fewer parameters and is easier to train, the performance needs to be improved. In future work, we will further optimize the system structure, so that the model can obtain rich semantic information characteristics.

## References

- Charlotte Ammer and Lea Grüner. 2020. Unituebingencl at SemEval-2020 task 7: Humor detection in news headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1060–1065.
- Rohan Badlani, Nishit Asnani, and Manan Rai. 2019. Disambiguating Sentiment: An Ensemble of Humour, Sarcasm, and Hate Speech Features for Sentiment Classification. *W-NUT 2019*, page 337.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. *Deep learning*, volume 1. MIT press Cambridge.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. SemEval-2020 task 7: Assessing humor in edited news headlines. *arXiv preprint arXiv:2008.00304*.
- Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv preprint arXiv:1909.11942*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 Task 7: HaHackathon, Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.
- Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Cernocky. 2011. Rnnlm-recurrent neural network language modeling toolkit. In *Proc. of the 2011 ASRU Workshop*, pages 196–201.
- Alex Morales and ChengXiang Zhai. 2017. Identifying humor in reviews using background text sources. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 492–501.
- Reynier Ortega-Bueno, Carlos E Muniz-Cuza, José E Medina Pagola, and Paolo Rosso. 2018. UO UPV: Deep linguistic humor detection in Spanish social media. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, pages 204–213.
- Bram Vanroy, Sofie Labat, Olha Kaminska, Els Lefever, and Véronique Hoste. 2020. Lt3 at SemEval-2020 task 7: Comparing feature-based and transformer-based approaches to detect funny headlines. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1033–1040. International Committee for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. *arXiv preprint arXiv:1909.00252*.

# UoR at SemEval-2021 Task 7: Utilizing Pre-trained DistilBERT Model and Multi-scale CNN for Humor Detection

Zehao Liu, Carl Haines, Huizhi Liang

University of Reading

White Knights, Berkshire, RG6 6AH

United Kingdom

z.liu3@pgr.reading.ac.uk, carl.haines@pgr.reading.ac.uk,  
huizhi.liang@reading.ac.uk

## Abstract

Humor detection is an interesting but difficult task in NLP. Humor might not be obvious in text because it may be embedded into context, hide behind the literal meaning of the phrase and require prior knowledge to understand. We explored different shallow and deep methods to create a humour detection classifier for task 7-1a. Models like Logistic Regression, LSTM, MLP, CNN were used, and pre-trained models like DistilBERT were introduced to generate accurate vector representation for textual data. We focused on applying a multi-scale strategy on modelling, and compared different models. Our best model is the DistilBERT+MultiScale CNN which used different sizes of CNN kernel to get multiple scales of features. This method achieved 93.7% F1-score and 92.1% accuracy on the test set.

## 1 Introduction

Humor detection is an interesting but difficult task in Natural language processing (NLP) and requires various techniques to understand the meaning of a sentence and identify humor. For example, humour by sarcasm can mean that a piece of text can have two very different meanings and the NLP algorithm needs to be able to understand which meaning is intended.

The aim of task 7-1a of SemEval 2021 (Meaney et al., 2021) was to address the challenge of classifying humour in text. Provided for this task was a dataset constructed of short phrases in English along with a label classifying whether or not each phrase is intended to be humorous. The labels have been obtained by surveying a group of people that represent a variety of genders, political stances and income levels. The given dataset includes training set with 8000 labeled sentences, a development set of 1000 sentences and a test set of 1000 sentences.

The dataset was made up of English phrases that were labeled by their intent to be humorous. This means the label annotators were not saying whether or not they found the text funny but whether the writer of the text intended it to be funny. The texts were predominantly one sentence long with a small proportion being two or three sentences. Each phrase was labeled for humour intent and, if humour was intended, then a rating was given for how funny it is. Also of the texts intended to be humorous, a label was given for whether it is offensive, and if so, how offensive it is.

The texts covered a range of types of jokes such as puns, sarcasm, dark jokes and “Dad” jokes. One example is “I never finish anything. I have a black belt in partial arts.” which contains a pun and is labeled as humorous.

Our best approach is DistilBERT+MultiScale CNN. It introduced a pre-trained DistilBERT model to extract textual features, and created a multi-scale CNN model for humour classification. First, the DistilBERT tokenizer generated a word token vector and an attention mask vector for each sentence. Then, we fed these vectors into a pre-trained DistilBERT model to get hidden features. After that, five CNN layers with different kernel sizes were used to get features of different scales. Each feature vector was subsequently concatenated together. Finally, the fused features were fed into dense layers in order to classify text into humorous or not humorous classes. It achieved 93.66% of F1 score and 92.10% of accuracy in using test set.

## 2 Related Work

We used four shallow models as a benchmark for our main approach. The first method was K-nearest neighbors (KNN) (Fix, 1951) in which an unlabeled query point is given the label of the majority of the K neighboring points. The second method

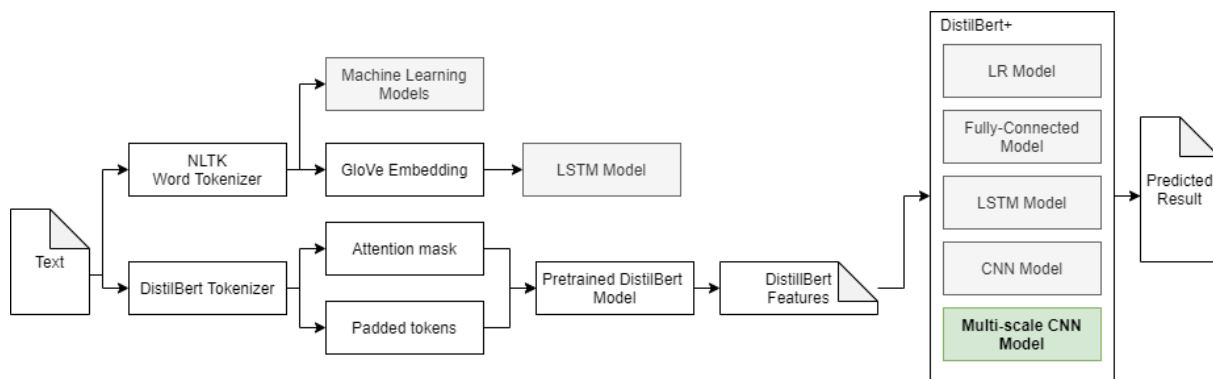


Figure 1: Pipeline of our experimental process

is Naive Bayes (Sammut C., 2011a) which utilizes Bayes rule together with a strong assumption that the attributes are conditionally independent. The third is a random forest (Sammut C., 2011b) which is an ensemble of decision trees trained on a bootstrap sample from the original dataset. The fourth method is the support vector machine (SVM) (Burges, 1998) which transforms the input data to a higher dimensional space and seeks to define a hyper-plane separating the classes. We also implemented voting to combine these methods where the prediction of the voting is simply the majority label predicted by the shallow models.

We also tried deep models. Long short-term memory (LSTM) is a very effective Recurrent neural networks for processing sequence data, which considers long and short term memory over time (Hochreiter and Schmidhuber, 1997). Convolutional Neural Network (CNN) uses a convolution kernel to extract hidden features from input, which is widely used in the field of computer vision (Le-Cun et al., 2010). 1-dimension CNN layer can fit with sequence data, so it can be used to extract textual features.

Transfer learning methods are widely used in the field of NLP. Pre-trained word embeddings are usually trained on unlabelled dataset and maps each word token into a fixed vector representing the meaning of the word in a hidden space. For example, Global Vectors for Word Representation (GloVe) is trained on a 6 billion word corpus using an unsupervised learning method in order to find word co-occurrence (Pennington et al., 2014).

Unlike conventional embeddings, contextualized embeddings dynamically map a word token into vectors based on the context using a pre-trained encoder. By extracting the features and adapting new data to the model, we can implement general down-

stream tasks based on the pre-trained model. BERT uses a deep Transformer as its encoder, and trains on language modelling tasks and next sentence prediction, which is often used in NLP with excellent performance (Tenney et al., 2019). DistilBERT uses knowledge distillation method to compress the model, which retains 97% of the performance of original BERT but is 60% faster (Sanh et al., 2020).

### 3 System Overview

In our BERT-based models, we used DistilBERT which is a light version of BERT to extract textual features. The BERT model is a transformer-based model, which is pretrained on vast amounts of textual data in language modelling tasks (Sanh et al., 2020). Therefore, the weights in the BERT model, which contains semantic information, can be used as contextualized embedding for general purposes. It can better represent textual data than a random tokenizer. Due to hardware limits, we only used the DistilBERT uncased base model in our system. It retains 97% performance of the original BERT but is 60% faster (Sanh et al., 2020). For comparison, we also used GloVe pretrained embedding to represent text and created a 2-layer LSTM model as our baseline. Shallow machine learning models were also explored for comparison. Figure 1 shows the pipeline of our experimental process.

Regarding feature extraction, we first used the DistilBERT tokenizer to vectorize the textual data then padded them into the same size. An attention mask was built to use binary vectors to differentiate padded zeros and word tokens. The vectors and mask were fed into the BERT model, and we stored the output of the last layer as the representation of text. Because the task was a text classification problem, we only used the “[CLS]” value in the text



representation for Logistic Regression (LR) and LSTM model, which was the first vector of each row. The length of each text representation was 768. For example, the sentence "Told my mom I hit 1200 Twitter..." was firstly tokenized into vectors "[101, 2409, 2026, 3566, 1045, 2718, 14840, 10474...]", and the [CLS] vector of DistilBERT output of this sentence was "[6.7492e-02, -1.6599e-01, 1.0417e-01, ...]", which had length of 768. For the Fully Connected (FC) model and all CNN models, we used the full output of the DistilBERT model as features, which were in shape of 136 x 768.

After extracting the features, we applied different models to predict the humor class. First, we implemented the LR model with default parameters. The package scikit-learn was used to build and train linear shallow models, such as LR, KNN, naive Bayes, random forest, and SVM. And then, we created a LSTM model. It consisted of two 32-node LSTM layers and a 32-node fully-connected layer. Also, we built a fully connected network, which had a 128-node dense layer and 64-node dense layer. In addition, we tried a CNN model which contained 2 64-node CNN-1D layer and a 128-node dense layer. For each network, we took the extracted features as input, and used a 1-node dense layer with sigmoid activation to collect output. In addition, dropout layers were used in each model to handle over-fitting problems.

For comparison, we also implemented a GloVe based LSTM. NLTK module was used to tokenize sentences and remove stop words and special characters. GloVe (Pennington et al., 2014) is pre-trained word vectors representation, which was used as the initial weights of the embedding layer. The GloVe+LSTM model used 50-dim embedding which connected to two 32-node LSTM layers. The output of LSTM was flattened and then fed into 16-node fully connected layer. And a 1-node dense layer with sigmoid activation was used to output probability of humorous class. It used the same model compiling parameters as the DistilBERT-based models.

Moreover, previous research proved that multiple scale of CNN layers can capture hidden features in different granularity, which improves performance (Cui et al., 2016; Yuan et al., 2018). Therefore, we build two multi-scale CNN models. The first model is DistilBERT+MultiScale CNN, Figure 2 shows its structure. It used five 64-node CNN-1D layers with different kernel size of [1, 2, 3, 4, 5].

These 5 layers could extract hidden information from the features in various granularity. Each CNN layer was then connected to a GlobalMaxPool1D layer to get a down-sampled representation in the shape of (batch size, 64). Also, dropout layers were applied on each output, and the 5 outputs were concatenated to a single vector in the shape of (batch size, 320). Subsequently, the combined vector was fed into a 512-node fully-connected layer. Rectify Linear Unit (ReLU) activation functions were applied to all CNN and fully-connected layers. Finally, a 1-node fully-connected layer with sigmoid activation function was added at the end of the network to collect the output value. We used 0.5 as a threshold to categorise the probability value into 0 (not humorous) and 1 (humorous) categories. In addition, we used "rmsprop" optimizer and binary cross entropy loss for stochastic gradient descent optimization.

In addition, we tried other multi-scale strategies, inspired by the deep multi-scale fusion hashing model (Nie et al., 2021). To differentiate two multi-scale models, we called this one MultiPool CNN. It used 5 different kernel size of MaxPooling layers to scale input features into different resolution. For each Maxpooling layer we set strides to 1 and used "valid" padding methods, and it connected to a 64-node CNN-1D layer with a kernel size of 1 and ReLu activation. Therefore, 5 CNN layers can extract features from different resolution of input. Similar to the multi-scale CNN, each output subsequently connected to a GlobalMaxPool1D layer and dropout layer. Finally, 5 different outputs were concatenated together and fed into a 512-node fully-connected layer. The output layer and compiling method are the same as the multi-scale CNN model.

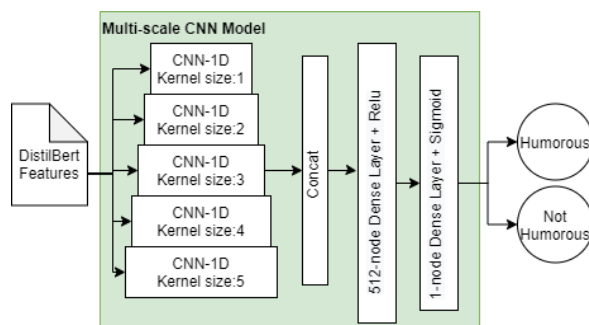


Figure 2: Structure of DistilBERT+Multi-Scale CNN

Model	F1@Dev	Accuracy@Dev	F1@Test	Accuracy@Test
<i>Official Baseline</i>	-	-	0.857	0.884
KNN	0.808	0.774	0.775	0.742
Naive Bayes	0.831	0.798	0.834	0.758
Random Forest	0.860	0.821	0.836	0.791
SVM	0.870	0.846	0.850	0.817
Voting Ensemble	0.853	0.821	0.817	0.774
GloVe+LSTM	0.888	0.850	0.885	0.853
DistilBERT+LSTM	0.884	0.842	0.896	0.862
DistilBERT+LR	0.898	0.867	0.904	0.880
DistilBERT+FC	0.904	0.877	0.925	0.907
DistilBERT+CNN	0.907	0.883	0.907	0.888
DistilBERT+MultiPool CNN	0.911	0.885	0.931	0.914
DistilBERT+MultiScale CNN	0.913	0.890	<b>0.937</b>	<b>0.921</b>

Table 1: Evaluation results of various implemented models on the dev set and test set (the gold data)

## 4 Experimental setup

We implemented our experiments on the university’s virtual machine, which has a python environment with a shared 16 GB Tesla v100 GPU. Tensorflow 2.0 and Keras were used to build neural networks. Python “transformers” module was used to import the DistilBERT model. “Pandas” and “Numpy” modules were utilised to manipulate data, and we used “scikit-learn” to import basic machine learning models and evaluation metrics.

Moreover, we used the training set to train our models and took 20% split of training set as validation set to tune hyper-parameter in development stage. The dev set was used to assess performance of models in evaluation stage, and we applied our models on test set in order to submit results. In data preprocessing, we load the data into a pandas data frame, and tried to remove all the tags and special characters in the text. After that, we used the DistilBERT tokenizer to vectorise and encode the text into the format that the DistilBERT model required. In all the models, we set 1 and 2 as random seeds for Numpy and TensorFlow respectively. Also, we used early stopping strategy when training models. All DistilBERT based neural network models were stopped at 23 epochs. Also, the batch size was set to 64.

Regarding evaluation, we evaluated our model on dev set and test set (gold data). F1 score and accuracy were used to evaluate performance. The accuracy score shows the ratio of correct prediction. Since the F1 score considers both precision and recall, which would be more informative metric, we selected our model based on the F1 score.

## 5 Results

We found that the features extracted by DistilBERT boosts the model performance significantly and it is difficult to generate a good results without using a pre-trained embedding or model. Due to hardware limits, we only tried the light version of BERT, and only implemented shallow layer models. Our official submission used DistilBERT+FC model, which achieved 92.41% F1-score and ranked 47th in task 7-1a. We modified our model in post-evaluation stage, and our best model is DistilBERT+Multi-scale CNN, which had 93.66% of F1 score and 92.10% of accuracy in using test set. The given official baseline is 85.7% of F1 score and 88.4% of accuracy. All of our DistilBERT based models and one GloVe embedding based model had better performance than the baseline. Detailed evaluation scores on dev set and test set were included in Table 1.

In comparison, the SVM model achieved an F1-score of 85.04% and an accuracy 81.70% on the test set, which is our best shallow model. However, all of these model generated scores lower than the given baseline.

Also, our GloVe-based LSTM model achieved 88.5% F1-score, and DistilBERT-based LSTM achieved 89.6%. It makes sense that transformer-based pretrained representation is better than traditional pretrained embedding. Due to efficiency considerations, we only used 32 nodes in the LSTM layer. So, these two models have poorer performance than other models using more nodes. Even the DistilBERT+LR model has higher F1 score (90.4%) and accuracy (88.0%). Surprisingly, a

	Not Humorous	Humorous
Not Humorous	337	48
Humorous	31	584

Table 2: Confusion matrix of DistilBERT+MultiScale CNN’s result

simple 2-layer fully connected network achieved 92.5% F1 on test set. But in the dev set, DistilBERT+CNN outperformed the DistilBERT+FC model. The CNN model had more stable performance, which achieved 90.7% F1 in both dev and test set. Because we used the same epochs for all models, CNN may converge better than the fully-connected model. The multi-scale strategy further improved the performance of the CNN model, which is our best model.

However, our best model still made mistakes on predicting humorous classes of a few sentences. An error analysis is helpful to understand the wrong predictions. In the prediction results of the DistilBERT+MultiScale CNN model, 79 out of 1000 sentences are incorrectly predicted. The Table 2 is a confusion matrix of the result. It shows that 48 predictions were false positive, which assigned a Not-Humorous sentence into the Humorous class. For example, ”I think in order to have a great business you have to like the product you’re selling more than the money you get.”, this sentence is misclassified as humorous. Also, 31 sentences are false negatives. Those sentences are labeled as humorous but ignored by our model. For example, ”If alcohol influences short-term memory, what does alcohol do?”, and ”And then there’s my dad...????”. Those sentences hide the humor within the context, which is hard for a model to detect. Some of sentences are also difficult for us to understand why they are humorous. Since the humor labels were created based on subjective judgement, even human beings would have diverse opinions and understanding.

## 6 Conclusions

To conclude, we explored various methods to build humour detection classifiers for task 7-1a. Models like Logistic Regression, LSTM, FC, CNN were used, and pre-trained models like DistilBERT were introduced to generate an accurate vector representation for textual data. Our best model is the DistilBERT+MultiScale CNN, which achieved 93.7% F1-score and 92.1% accuracy on the test set. We

focused on applying multi-scale strategy on modelling, and compared different models. And our results shows that CNN are more suitable for this task than LSTM FC and other shallow models. Also, we found that pre-trained embeddings, weights or representations are crucial for our model performance. We only explored multi-scale from the wide dimension, this strategy can also be used in deep dimension. In the future, a deeper network with more nodes can be explored and the full version of BERT model can be exploited.

## References

- Christopher J.C. Burges. 1998. [A tutorial on support vector machines for pattern recognition](#). *Data Mining and Knowledge Discovery*, 2(2):121–167.
- Zhicheng Cui, Wenlin Chen, and Yixin Chen. 2016. [Multi-scale convolutional neural networks for time series classification](#).
- Joseph L. Fix, Evelyn; Hodges. 1951. Discriminatory analysis. nonparametric discrimination: Consistency properties. *USAF School of Aviation Medicine, Randolph Field, Texas*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long short-term memory](#). *Neural Comput.*, 9(8):1735–1780.
- Yann LeCun, Koray Kavukcuoglu, and Clément Faret. 2010. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- X. Nie, B. Wang, J. Li, F. Hao, M. Jian, and Y. Yin. 2021. [Deep multiscale fusion hashing for cross-modal retrieval](#). *IEEE Transactions on Circuits and Systems for Video Technology*, 31(1):401–410.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. [Glove: Global vectors for word representation](#). In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- Webb G.I. Sammut C. 2011a. [Naïve Bayes](#). In: *Encyclopedia of Machine Learning*. Springer.
- Webb G.I. Sammut C. 2011b. [Random forests](#). In: *Encyclopedia of Machine Learning*. Springer.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.](#)

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. [What do you learn from context? probing for sentence structure in contextualized word representations.](#)

Q. Yuan, Y. Wei, X. Meng, H. Shen, and L. Zhang. 2018. [A multiscale and multidepth convolutional neural network for remote sensing imagery pan-sharpening.](#) *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(3):978–989.

# TECHSSN at SemEval-2021 Task 7: Humor and Offense detection and classification using ColBERT embeddings

Rajalakshmi Sivanaiah, Angel Deborah S, S Milton Rajendram, Mirnalinee T T, Abrit Pal Singh, Aviansh Gupta, Ayush Nanda

Department of Computer Science and Engineering  
Sri Sivasubramaniya Nadar College of Engineering  
Chennai 603 110, Tamil Nadu, India

{rajalakshmis, angeldeborahs, miltonrs, mirnalineett}@ssn.edu.in  
{abritpal18007, aviansh18028, ayush18031}@cse.ssn.edu.in

## Abstract

This paper describes the system used for detecting humor in text. The system developed by the team TECHSSN uses binary classification techniques to classify the text. The data undergoes preprocessing and is given to ColBERT (Contextualized Late Interaction over BERT), a modification of Bidirectional Encoder Representations from Transformers (BERT). The model is re-trained and the weights are learned for the dataset. This system was developed for the task 7 of the competition, SemEval 2021.

## 1 Introduction

Natural language processing faces the challenges working with humor as it is a highly subjective phenomena and the age, gender and socio-economic status are known to have an impact on the perception of the joke. It usually involves multiple word senses and cultural knowledge to appreciate humor to its best. Now a days chatbots and virtual assistants require automated humor detection systems for a better interaction with the user by understanding what a human-like approach to humor is. It is crucial to understand the real motive of the user and provide appropriate answer to have a better experience of the user with the virtual assistants (Chen and Soo, 2018). Based on the general linguistic structure of humor, we propose an approach for detecting humor in short texts using ColBERT in this paper.

We have developed a system in the name of TechSSN for previous SemEval tasks (Sivanaiah et al., 2020; Logesh et al., 2019) for offensive language detection using various machine learning and deep learning networks. In SemEval 2021, we participated in subtask-1a and

1c of humor and offensiveness detection in task 7- HaHackathon (Meaney et al., 2021).

## 2 Related Work

Continuous research is going on in this field of humor detection and the systems are getting better every year. De Oliveira and Rodrigo (2015) developed a model for humor detection in Yelp reviews and used convolutional networks with a maximum of 81.57% accuracy.

The system developed by Ortega-Bueno et al. (2018) used UO\_UPV, a Attention-based Long Short-Term Memory Network. The model consists of a Bidirectional LSTM neural network with an attention mechanism that allows to estimate the importance of each word and then, this context vector is used with another LSTM model to estimate whether the tweet is humorous or not. The F1 score for this system is approximately 0.78 with accuracy, 0.84.

Mao and Liu (2019) developed a system with BERT, a multi-layer bidirectional transformer encoder which can help to learn deep bidirectional representations, and the pretrained model is fine-tuned on training data. Their best F1 Score on the test set is 0.784.

Risch et al. (2020) explained the need and various methods used for offensive language detection. BERT model is used with transfer learning for the offensiveness detection by (Liu et al., 2019) with F1 score as 0.8286 and accuracy as 0.8628.

## 3 Methodology

ColBERT base model is chosen for humor text classification which has 8 layers with the last layer's activation function as the sigmoid func-

tion, as it is performing binary classification. The remaining layers have ReLU activation function.

### 3.1 Model Architecture

This classification model uses a separate line of hidden layers especially designed to extract features from each sentence. The used model is a neural network that includes two parallel lines of hidden layers: One to view text as a whole and another to view each sentence separately. Figure 1 displays the architecture of the proposed method.

First, to assess each sentence separately and extract numerical features, the sentences are separated and are tokenized individually. To prepare these textual parts as proper numerical inputs for the neural network, they are encoded using BERT sentence embedding (Devlin et al., 2018). This step is performed individually on each sentence and also on the whole text (shown in Figure 1). As we get the BERT sentence embedding for each sentence, they are fed into the parallel hidden layers of the neural network to extract mid-level features for each sentence (could be related to context, type of sentence, etc). The vector size obtained from this layer for each sentence is 20.

While the main idea is to detect relationships between sentences (especially with punchline), it is also essential to find out the word-level connections in the whole text (such as synonyms and antonyms). Identifying this relation will have meaningful impacts in determining congruity of the text. Similar to the previous step, we feed BERT sentence embedding for the whole text into hidden layers of the neural network. The vector size is 60. Finally, there are three sequential layers in the neural network model. These final layers combine the output of all previous lines of hidden layers in order to return the final output. These final layers are used to determine the congruity of sentences and detect the transformation of reader’s viewpoint after reading the punchline.

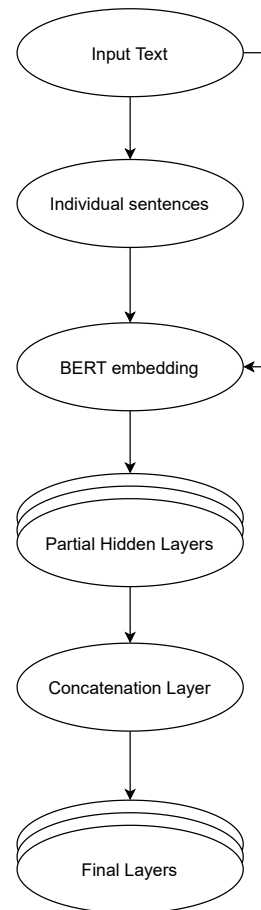


Figure 1: Model Architecture

### 3.2 Dataset Collection

For building the model we have used the training dataset provided by the organizers of the Hahackathon - (Meaney et al., 2021). This dataset has 8000 instances of which 4932 belong to humor class and 3068 belong to non-humor class. Out of the 4932 humor instances, 2465 are controversial and 2467 are not controversial. Each instance in the dataset has four features: `is_humor`, `humor_rating`, `humor_controversy` and `offense_rating`. We have trained our model for the classification features only; `is_humor` and `humor_controversy`. Both features have binary values.

### 3.3 Data Preprocessing and Tokenization

For data preprocessing the function `convert_to_transformer_inputs` is used to convert the tokenized input into ids, masks and segments for the transformer. The tokenization of the dataframe columns is done by the function

*compute\_input\_arrays*. BERT Tokenizer pre-trained on the ‘BERT-base-uncased’ model is used for tokenization. The maximum sequence length for reading the data is set as 200 that will be used as the input to BERT.

### 3.4 Model Creation

For the model we have used BERT-base. We use a function *create\_model* that has the architecture which is used to fine tune BERT to our chosen dataset, and we compute the competition metric for the validation set with the help of the function spearman rank correlation coefficient.

### 3.5 Training, Cross Validation and Testing

The model is trained with cross validation for 3 epochs with a learning rate of  $3e-5$  and the size of each batch is 6. As we have performed binary classification for the humor detection task, we have set the loss function as a simple binary crossentropy.

## 4 Results and Discussion

The test dataset given by SemEval organisers (gold-test-27446.csv) was tested on different models and the F1 score for all these models are discussed in the following sections.

### 4.1 ColBERT

We have used Contextualized Late Interaction over BERT (ColBERT) (Khattab and Zaharia, 2020). ColBERT differs by providing a late interaction architecture that independently encodes the query and the document using BERT. It then uses a powerful interaction step that analyze their similarity in a fine grained mode. Eventhough the interaction learning is delayed it also maintains this fine-granular interaction in a better manner. ColBERT can leverage the expressiveness of deep language models and simultaneously gaining the ability to pre-compute document representations in an of-line structure. This will speed up query processing since the representations are learnt in offline. Beyond reducing the cost of re-ranking the documents retrieved by a traditional model,

ColBERT’s pruning-friendly interaction mechanism enables leveraging vector-similarity indexes for end-to-end retrieval directly from a large document collection.

### 4.2 Decision Tree

Decision tree (DT) is a popular supervised technique used for classification problems. It identifies the relation between the features and form the set of rules that can be used to classify the given data into any one of the class labels. The method uses the train dataset to generate branch-like segments that construct an inverted tree with a root node, internal nodes, and leaf nodes.

### 4.3 SVM

Support Vector Machine (SVM) is a supervised machine learning technique used for both classification or regression problems. It uses the concepts of separating the classes using maximal margin hyperplane. We fed pre-processed data into Support Vector Classifier (SVC) with Gaussian Radial Basis (RBF) kernel for training and testing.

Table 1 shows the F1 score and accuracy for the best, baseline and our approach.

No.	System	F1	Accuracy
1	Best Approach	0.982	0.9854
2	Baseline	0.857	0.848
3	Our Approach	0.884	0.9081

Table 1: Official results of the humor detection task

Figure 2 shows the F1 score and the accuracy for the various models we tested for humor detection. ColBERT model provides better accuracy when compared to decision tree and support vector machine models.

No.	Model	F1	Accuracy
1	ColBERT	0.884	0.9081
2	SVM	0.747	0.617
3	Decision Tree	0.736	0.62

Table 2: Results for various models used for humor detection

The humor testset contains 1000 instances of which 615 are humorous and 385 are not. The confusion matrix for ColBERT model is given in Table 3.

		Actual	
		Humor	Not
Predicted	Humor	553	30
	Not	62	355
	Total	615	385

Table 3: Confusion matrix of ColBERT for humor detection

There are 279 offensive texts and 336 non offensive texts in 615 humor instances. Table 4 shows the results for ColBERT and SVM model used for humor controversy or offensiveness detection.

No.	Model	F1	Accuracy
1	ColBERT	0.53	0.56
2	SVM	0.487	0.530

Table 4: Results for various models used for humor controversy detection

## 5 Conclusion

There is a lot of demand for automatic humor detecting systems in the market as it can be used in chatbots and AI assistants to achieve human-like experience while talking to a machine. SemEval-2021 task 7 involves a subtask-1a as identifying humor in text. A modification of the BERT called ColBERT is used to classify such text sentences into humorous or not. ColBERT is a 8-layer model with 110M parameters outperforms the machine learning models we tested with a large margin, showing the importance of utilizing linguistic structure. The preprocessing techniques can be enhanced to get better accuracy.

## References

Peng-Yu Chen and Von-Wun Soo. 2018. Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of*

*the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117.

Luke De Oliveira and Alfredo Láinez Rodrigo. 2015. Humor detection in yelp reviews. *Retrieved on December*, 15:2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Xiaochao Fan, Hongfei Lin, Liang Yang, Yufeng Diao, Chen Shen, Yonghe Chu, and Yanbo Zou. 2020. Humor detection via an internal and external neural network. *Neurocomputing*, 394:105–111.

Ashraf Kamal and Muhammad Abulaish. 2019. Self-deprecating humor detection: A machine learning approach. In *International Conference of the Pacific Association for Computational Linguistics*, pages 483–494. Springer.

Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48.

Ping Liu, Wen Li, and Liang Zou. 2019. NULI at SemEval-2019 task 6: Transfer learning for offensive language detection using bidirectional transformers. In *Proceedings of the 13th international workshop on semantic evaluation*, pages 87–91.

B Logesh, S Harshini, B Geetika, S Dyaneswaran, S Rajalakshmi, Angel Suseelan, S Milton Rajendram, and TT Mirmalinee. 2019. TECHSSN at SemEval-2019 Task 6: Identifying and categorizing offensive language in tweets using deep neural networks. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 753–758.

Jihang Mao and Wanli Liu. 2019. A BERT-based approach for automatic humor detection and scoring. In *IberLEF@ SEPLN*, pages 197–202.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 Task 7, HaHackathon, Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Reynier Ortega-Bueno, Carlos E Muniz-Cuza, José E Medina Pagola, and Paolo Rosso. 2018. UO UPV: Deep linguistic humor detection in spanish social media. In *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, pages 204–213.



Julian Risch, Robin Ruff, and Ralf Krestel. 2020. Offensive language detection explained. In *Proceedings of the Second Workshop on Trolling, Aggression and Cyberbullying*, pages 137–143.

Sushmitha Reddy Sane, Suraj Tripathi, Koushik Reddy Sane, and Radhika Mamidi. 2019. Deep learning techniques for humor detection in Hindi-English code-mixed tweets. In *Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 57–61.

Rajalakshmi Sivanaiah, Angel Suseelan, S Milton Rajendram, and Minalinee TT. 2020. TECHSSN at SemEval-2020 Task 12: Offensive language detection using BERT embeddings. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 2190–2196.

# Amherst685 at SemEval-2021 Task 7: Joint Modeling of Classification and Regression for Humor and Offense

Brian Zyllich   Akshay Gugnani   Gabriel Brookman   Nicholas Samoray

University of Massachusetts Amherst

{bzylich, agugnani, gbrookman, nsamoray}@umass.edu

## Abstract

This paper describes our submission to the SemEval’21: Task 7- HaHackathon: Detecting and Rating Humor and Offense. In this challenge, we explore intermediate finetuning, backtranslation augmentation, multitask learning, and ensembling of different language models. Curiously, intermediate finetuning and backtranslation do not improve performance, while multitask learning and ensembling do improve performance. We explore why intermediate finetuning and backtranslation do not provide the same benefit as on other natural language processing tasks and offer insight into the errors that our model makes. Our best performing system ranks 7th on Task 1b with an RMSE of 0.5339.

## 1 Introduction

With the advancement in deep learning methods, NLP tasks like sentiment analysis and opinion mining have achieved high accuracy, however detection of some salient forms of figurative language such as humor remain difficult tasks.

Being able to infer humor and offense with a high accuracy can help improve and lead to better performance on downstream applications, such as content moderation, sentiment analysis, etc. This would be useful for various downstream applications, such as understanding tweets, reviews and feedback. However, humor detection is not trivial.

What makes identifying humor hard? Humor can consist of styles ranging from sarcastic to slapstick comedy, and it factors in both individual preferences and underlying cultures. Context, sounds, and vision or any combination of these can be key in building to a punchline (Cai, 2019). Humor appreciation is also highly subjective, as age, gender, and socio-economic status often impact the perception of a joke. Meaney (2020) identify 3 ways that classification of humor is difficult:

- (1) Humor can differ between cultures,
- (2) Humor can also differ within cultures, and
- (3) Humor differs within the same person.

Our contributions are as follows: (1) we explore whether intermediate finetuning on other humor and offense datasets is helpful for this task, (2) we seek to identify if backtranslation augmentation is useful for humor detection, (3) we show that multitask learning is helpful when classifying and scoring both humor and offense, and (4) we find that ensembling different language models leads to improved results on some tasks. The code for our experiments is available at our Github repository<sup>1</sup>.

## 2 Related Work

The challenge of humor detection has gained traction since 2017. Meaney (2020), explains in their proposal that prior work has explored humor detection as an objective task, averaging all annotations for a joke, to produce a single classification or rating. This treats humor as an objective concept, which is not the case. This motivated their challenge for SemEval’21 (Meaney et al., 2021) to explore these dimensions of humor and offense. We discuss some of the previous efforts to explore humor and offensiveness in the following paragraphs.

Badlani et al. (2019) explains how text in reviews is quite complex as they can be sarcastic, humorous, or hateful. An ordinary sentiment analysis would fail to perform well in such cases. They first extract features pertaining to sarcasm, humor, hate speech, and sentiment, and then use these features to inform sentiment classification. Their work is quite sensitive to catching negative sentiment, however, it does not do as well when sentiment changes halfway through the text. It also does not address the subjectivity of humor.

<sup>1</sup><https://github.com/bzylich/humor-by-demographic>

ColBERT (Annamoradnejad, 2020) is among the first to use BERT (Devlin et al., 2018) for humor detection, reaching 98% classification accuracy and outperforming variants using recurrent neural networks and convolutional neural networks. Mao and Liu (2019) is another work that uses BERT to classify if a tweet is a joke or not and predict how humorous the tweet is. The work of Weller and Seppi (2019) explores extending humor detection capability by trying to assess whether or not a joke is humorous. They use transformers to identify humorous jokes based on ratings from Reddit pages, reaching human-level performance.

Earlier work, like (Donahue et al., 2017), use recurrent deep learning methods with dense embeddings to predict humorous tweets. In order to factor both meaning and sound in their analysis, they use GloVe embeddings combined with a novel phonetic representation as input to an LSTM.

Hossain et al. (2020) hosted the SemEval’20 event for humor detection in news headlines. The event challenged participants to classify whether an original headline or an altered headline is funnier and rate the funniness of the edited headline on a 0-3 humor scale. The winning teams (Morishita et al., 2020) combined the predictions of several models using sentence pair regression and ensembled the pre-trained language models BERT, GPT-2, (Radford et al., 2019) RoBERTa, (Liu et al., 2019), XLNet (Yang et al., 2019), Transformer-XL, and XLM (Dai et al., 2019) to form the final prediction.

Similar to humor detection, there has been some work to explore offense in text. SemEval ’19 had a task (Zampieri et al., 2019), aimed at identifying and categorizing offensive language in social media. The top performing teams used ensembles of random forest, linear models, recurrent networks, and pretrained transformer language models.

Our work is motivated by the SemEval challenges which encourage interesting techniques to handle multiple word senses, cultural knowledge, and pragmatic competence. Through this challenge, we try to detect humor and explore the subjectivity of humor appreciation with a controversy score to examine the variance in humor ratings for each different text.

## 3 Dataset

### 3.1 Data

We use three types of datasets in this work: the HaHackathon competition dataset (Meaney et al.,

2021), datasets for offensive text detection, and datasets for humor detection. We describe each of these in the following subsections.

#### 3.1.1 HaHackathon Competition Dataset

The training dataset consists of 8000 texts (Meaney et al., 2021) and four subtasks: humor classification, humor rating, humor controversy, and offense rating. For our initial experiments we created a randomized 90-10 train-development split of 7200 training examples and 800 sentences for model development. In addition, the competition has its own development dataset of 1000 texts. The labels for this dataset were released during the final stage of the evaluation. The gold-standard test dataset (Meaney et al., 2021) is the one we use for our system results.

#### 3.1.2 Humor Datasets

For the humor component of the competition, we use two datasets for intermediate finetuning: 200k Short Texts for Humor Detection (Annamoradnejad, 2020) and a self-compiled dataset of jokes and other texts scraped from Reddit.

The 200k Short Texts for Humor Detection dataset consists of 200,000 short text snippets, each labeled as either humorous or not humorous, with an even split between the two classes. The non-humorous texts are news headlines from the Huffington Post while the humorous texts were taken from Reddit communities such as /r/jokes and /r/cleanjokes.

The other dataset was one which we compiled ourselves, consisting of 200,000 snippets of text scraped from various reddit communities. This was primarily to address shortcomings we noticed in the 200k Short Texts dataset; namely the limited range of lengths for jokes and the singular source for the negative examples. For the positive examples of humor, we scraped the /r/jokes subreddit. For negative examples, we scraped subreddits such as /r/reddit.com and /r/worldnews, which offer more variability in the types of non-humorous texts than just news headlines from one news website.

#### 3.1.3 Offense Datasets

The Offensive Language Identification Dataset (OLID) (Zampieri et al., 2019) is one dataset for identifying offensive language in texts, specifically tweets. It contains 14,200 tweets as well as binary annotations indicating whether or not they are offensive.

Another similar dataset is the Hate Speech and Offensive Language dataset (Davidson et al., 2017). This dataset consists of 26,953 tweets as well as labels corresponding to how many crowdflower users labeled them as hateful and/or offensive.

Together, we experiment with intermediate finetuning on both of these datasets in the hope that it would provide some benefit for offensive language detection in task 2.

## 4 System Overview

As a starting point, we use the HuggingFace Transformers library<sup>2</sup>, along with their large collection of pretrained language models. From there, we finetune these transformers on the competition training dataset during development and the combined training and development datasets for the final evaluation phase.

Building on this basic paradigm, we experiment in four different ways with the goal of improving model performance: (1) using various datasets for intermediate finetuning, (2) using backtranslation to expand the competition datasets, (3) training multitask models to predict all labels simultaneously, and (4) ensembling predictions using different pretrained language models as starting points.

### 4.1 Intermediate Finetuning

We tried using intermediate finetuning on larger datasets for humor detection and offensive language detection in the hope that these larger datasets would provide a better starting point for training on the competition data, which is relatively small at just 8000 texts.

We perform intermediate finetuning in the same manner as the previously described basic transfer learning setup, and then we perform an additional transfer from the intermediate task to the competition task. For intermediate tasks we try using the two offensive language identification datasets previously mentioned, and for humor we tried using the 200k humor dataset and the Reddit dataset we collected.

We also try using ColBERT (Annamoradnejad, 2020), a pretrained BERT model that has been finetuned for humor prediction, as a starting point for intermediate finetuning and as a pretrained language model for the standard transfer approach.

<sup>2</sup><https://huggingface.co/transformers>

### 4.2 Backtranslation Augmentation

As another method of expanding the training dataset and introducing variation, we use backtranslation to create paraphrases of the texts in the dataset. These paraphrases are generated by first translating the text into a different language using the Google Translate python library<sup>3</sup>, and then translating the text back into the original language, usually with some small variations in the wording or sentence structure. This augmentation is useful in other tasks, but it was not clear whether the backtranslation would preserve humor, as some humor is generated based on the specific words or sounds used (e.g. puns).

### 4.3 Multitask Models

Initially, we train one model for each task or subtask. We also try training one model to learn to predict the labels associated with all four tasks or subtasks at the same time. To accomplish this, we attach four different heads on top of the final transformer outputs. Each prediction head consists of two fully-connected feed-forward layers matching the dimensionality of the transformer layers used by the pretrained language model, and an output layer that produces a single regression score or binary probabilities depending on the task.

### 4.4 Ensembling Model Predictions

We did most of our development with DistilBERT (unless otherwise specified) because it is relatively fast to train and run, allowing us to iterate more rapidly. We hypothesized that different pretrained language models would have different strengths and weaknesses when finetuned due to the different pretraining data used and the different model architectures. By ensembling (+Ens) many language models together, we might then counterbalance the weaknesses of individual models to improve overall performance.

Ultimately, we experimented with 6 model variants: “distilbert-base-uncased”, “distilroberta-base”, “bert-base-uncased”, “roberta-base”, “bert-large-uncased-whole-word-masking”, and “roberta-large” pretrained language models from the HuggingFace Transformers library. To get the predictions for each model, we average together the predictions from 5 different random restarts to mitigate the effect of variance induced by the random initialization. To ensemble the different models

<sup>3</sup><https://pypi.org/project/googletrans/>

together, we simply averaged the predictions from each model together to form the final predictions, taking the argmax of the averaged probabilities for classification tasks.

For a slightly more advanced ensembling method, for each task we select the models to average together by trying all possible combinations and selecting the combination that leads to the best performance on the development dataset (+Ens-Best). Then, we use the same model combinations to generate predictions to submit to the competition leaderboard.

## 4.5 Experimental Setup

To facilitate transfer learning on top of the original language model, we add two linear layers for each task on top of the CLS token of the transformer. The first linear layer has the same dimension as the language model. After the first linear layer, we use ReLU activation, and the second layer produces the prediction (classification or regression depending on the task). For training each model we use the same hyperparameters: a batch size of 10, a learning rate of  $5e-5$ , 3 epochs, 500 warmup steps, and a weight decay of  $0.01^4$ . During intermediate finetuning, we transfer all weights from each prior finetuning step and all weights remain trainable at each step.

## 4.6 Results

We find that intermediate finetuning on other datasets for humor and offense identification do not improve performance. Similarly, using ColBERT as a starting point does not outperform other pretrained language models. This may be due to differences in how these datasets were sourced, and more analysis is provided in section 5. We also find that backtranslation augmentation is not helpful for humor detection, likely because it does not always preserve humor. While not beneficial, it is noteworthy as it is not clear whether prior work has explored backtranslation for expanding humor datasets, and this work suggests that backtranslation should not be used in contexts such as humor which are highly dependent on the specific words and sounds in a text.

Here, we show an example where backtranslation does not preserve humor since the word *imaginary* is a key part of the joke and it is substituted during the translation process:

<sup>4</sup><https://github.com/bzylich/humor-by-demographic>

- Original: My girlfriend is like the square root of -100. She's a 10 but she's imaginary.
- Backtranslation: My girlfriend is like the square root of -100. She is 10 but she is fantastic.

While many translations do not preserve humor, some translations do successfully preserve humor while introducing some word variation into the text:

- Original: My father doesn't trust anyone. In fact he has a saying... But he won't tell me.
- Backtranslation: My dad doesn't trust anyone he has a saying ... but he doesn't tell me.

Next, multitask learning improves performance over training models for each task individually, especially for some tasks such as humor rating prediction and humor controversy prediction. Finally, ensembling different pretrained language models together leads to an increase in performance, suggesting that these models complement each other by mitigating other models' weaknesses. Table 1 shows which submissions perform best on each individual task.

## 5 Error analysis

One of the possible sources of confusion and bias in our model seemed to be centered around atypical punctuation such as question marks and exclamation marks. For example, when a question mark was placed in the middle of a sentence, the model often erroneously labels it humorous regardless of the actual content. When manually reviewing the data, we found that the vast majority of texts that contain a mid-text question mark are humorous due to their setup and punchline structure. Without balancing with negative examples with similar structures, the model can become reliant on punctuation structure rather than the actual relationship between the words.

Another driver of error seems to be the actual source of the competition dataset. Through further analysis, we found that the vast majority of the negative examples seemed to be sourced from tweets. This can be seen in the length distribution of the dataset; there is a sharp cutoff around 140 characters, which used to be the maximum length for a tweet. However none of the other datasets we found or compiled ourselves (for Humor tasks)

Approach	Humor F1 / Acc	Humor RMSE	Controversy F1 / Acc	Offense RMSE
RoBERTa-Large Multitask	<b>0.9510 / 0.9604</b>	<b>0.5339</b>	0.5220 / <b>0.4842</b>	0.4564
Multitask +Ens=6LM	0.9460 / 0.9565	0.5457	<b>0.5528</b> / 0.4841	0.4606
Multitask +Ens-Best=6LM	<b>0.9510 / 0.9604</b>	0.5411	0.5415 / 0.4659	<b>0.4530</b>

Table 1: Competition Results

contained information from Twitter specifically, almost all relied heavily on news headlines instead. When pulling individual examples, we found that tweets tended to use more colloquial language, with a greater variety of punctuation, vocabulary, and capitalization when compared to news headlines.

One final potential driver of error were song lyrics and quotes. There were a proportionally large number of movie, song, and TV show quotes used in the dataset, by a rough estimate based on sampling, approximately 5% of the example fell in one of those categories. Our model was often able to differentiate between these quotes, though it was not something that was found in our own custom datasets.

After performing this deep dive analysis on our results, and seeing the various areas of where our model got confused, we believe that the primary reason our models did worse with the inclusion of extra datasets was due to the source of our data. The wider range of punctuation, capitalization, and vocabulary expressed in twitter posts was not well captured by utilizing news headlines as a negative source, and thus likely allow our model to use syntax and punctuation structure as a substitute for the actual substance of the text.

## 6 Conclusion

In this competition, we explored the use of intermediate finetuning, backtranslation augmentation, multitask learning, and ensembling of different pretrained models. Unlike in many natural language processing tasks, intermediate finetuning on other related datasets provided no benefit in this task, perhaps because prior datasets used non-humorous texts that were much easier to identify. Next, although backtranslation augmentation did not improve performance, this is still a noteworthy result because it indicates that humor is likely not preserved through paraphrasing. Finally, multitask learning across humor and offense detection, as well as ensembling of different pretrained language models improved overall performance.

## References

- Issa Annamoradnejad. 2020. ColBERT: Using BERT sentence embedding for humor detection. *arXiv preprint arXiv:2004.12765*.
- Rohan Badlani, Nishit Asnani, and Manan Rai. 2019. Disambiguating sentiment: An ensemble of humour, sarcasm, and hate speech features for sentiment classification. *W-NUT 2019*, page 337.
- Fangyu Cai. 2019. [Does ai get the joke?](#)
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media, ICWSM '17*, pages 512–515.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- David Donahue, Alexey Romanov, and Anna Rumshisky. 2017. [HumorHawk at SemEval-2017 task 6: Mixing meaning and sound for humor recognition](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 98–102, Vancouver, Canada. Association for Computational Linguistics.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. Semeval-2020 task 7: Assessing humor in edited news headlines. *arXiv preprint arXiv:2008.00304*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Jihang Mao and Wanli Liu. 2019. A BERT-based approach for automatic humor detection and scoring. In *IberLEF@ SEPLN*, pages 197–202.
- J. A. Meaney. 2020. Crossing the line: Where do demographic variables fit into humor detection? In

*Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 176–181.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 task 7, HaHackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Terufumi Morishita, Gaku Morio, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. Hitachi at SemEval-2020 task 7: Stacking at scale with heterogeneous language models for humour recognition. In *14th International Workshop on Semantic Evaluations (SemEval-2020)*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Orion Weller and Kevin Seppi. 2019. Humor detection: A transformer gets the last laugh. *arXiv preprint arXiv:1909.00252*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. SemEval-2019 task 6: Identifying and categorizing offensive language in social media (OffensEval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

# DuluthNLP at SemEval-2021 Task 7: Fine-Tuning RoBERTa Model for Humor Detection and Offense Rating

Samuel Akrah

Department of Computer Science  
University of Minnesota Duluth  
Duluth, MN 55812 USA  
akrah001@d.umn.edu

## Abstract

This paper presents the DuluthNLP submission to Task 7 of the SemEval 2021 competition on Detecting and Rating Humor and Offense. In it, we explain the approach used to train the model together with the process of fine-tuning our model in getting the results. We focus on humor detection, rating, and offense rating, representing three out of the four subtasks that were provided. We show that optimizing hyper-parameters for learning rate, batch size and number of epochs can increase the accuracy and F1 score for humor detection.

## 1 Introduction

Humor detection poses a challenge to humans, not least because of the mix of irony, sarcasm, and puns which underlie humor. To understand the funniness of humor requires a certain grasp of context, culture and, for some, even country.

If rating the funniness of humor is any challenge, ranking its offensiveness is even more so, especially when doing so requires an appreciation of the sensibilities of the humor target – whether race, religion, and/or gender – and, in most cases, context.

It is little wonder humor detection has been central to NLP tasks in the past few years. The last few SemEvals have featured tasks focused exclusively on either detecting the funniness of humor (Hossain et al., 2020; Van Hee et al., 2018; Potash et al., 2017) or detecting offense (Zampieri et al., 2019). What SemEval-2021 task 7 seeks to do is combine the detection of both humor and offense for a given corpus.

Our approach uses pretrained RoBERTa model (Liu et al., 2019b) trained on a RoBERTa classifier implemented by HuggingFace (Wolf et al., 2019). The intuition here is that RoBERTa model achieves state of the art performance for tasks requiring contextual information. Our goal is to measure the

effect of varying three hyperparameters – batch size, learning rate, epoch size - whilst maintaining default values for others. Our results show that varying the three hyperparameters can increase performance for humor detection, humor ranking, and offense rating. The codebase for our participation of this SemEval Task is available on github <sup>1</sup>

## 2 Related Work

Earlier works on humor detection and rating showed modest gains. With the advent of attention mechanism (Vaswani et al., 2017), though, and the transformer model a few years later (Dai et al., 2019), not only has interest in the NLP community on humor detection has soared, but performance on humor and offense detection has increased (Weller and Seppi, 2019).

This has been particularly so in the last few years, where humor detection and offense rating have been featured in some of SemEval tasks. SemEval-2019 Task 6 on offense rating (Zampieri et al., 2019) attracted 800 participants and 115 submissions, the interest prompting a second SemEval-2020 Task 12 the following year (Zampieri et al., 2020). Around the same period, humor rating have attracted similar interest (Hossain et al., 2020) in SemEval Tasks. To the best of our knowledge, however, SemEval-2021 Task 7 (Meaney et al., 2021) is the first to measure humor and offense for a given task.

Most of the winning teams (Morishita et al., 2020; Rozental and Biton, 2019; Wiedemann et al., 2020), for both humor and offense rating alike, implemented BERT and its variants, including Albert and RoBERTa, in their model, an approach that tended to yield the best results. And more often than not, the teams exploit ensembles of BERT,

<sup>1</sup><https://github.com/akrahdan/SemEval2021>



GPT-2, RoBERTa and their variants (Morishita et al., 2020), whilst others stick to a single pre-trained model.

Our approach in this task is to use RoBERTa model, an approach that will fine-tune a select number of hyperparameters and to measure the model performance for every change of hyperparameter set.

### 3 System overview

In this section, we review our system’s adoption of the pretrained RoBERTa model (Liu et al., 2019a) for SemEval tasks. We also describe the Bayesian hyperparameter optimization technique, which we used in our hyperparameter sweeps for selecting optimal values for learning rate, batch size, and epoch cycles.

#### 3.1 Model description

Our system’s adoption of RoBERTa model is based on its ability to achieve state-of-the-art performance for most NLP tasks with minimal effort, including, in our case, humor detection. The RoBERTa model, itself a re-implementation of BERT (Devlin et al., 2019), is first pre-trained on unlabeled text corpus and subsequently fine-tuned on downstream tasks with labeled data.

The RoBERTa model is a significant improvement over the BERT model, and it differs from BERT for its usage of dynamic masking for training, Next Sentence Prediction (NSP), and a larger mini-batch size (which, it has been observed, correlates with performance (Liu et al., 2019a)).

Again, the RoBERTa model outperforms BERT for its size and diversity of data used in pretraining, with its 160GB of training data drawn from multiple sources compared to BERT’s 16GB of training data.

Our system implements the RoBERTa model with a classification layer on top using HuggingFace transformer model<sup>2</sup>.

We adopt Bayesian optimization to automate the selection of optimal hyperparameter values for the training and evaluation of the three Subtasks. Details of the Bayesian optimization method are found in Appendix A

#### 3.2 Sweeps

We use Bayesian optimization to run hyperparameter sweeps for our model, but not before manually

<sup>2</sup><https://huggingface.co/transformers/>

selecting a sensible list of hyperparameter values in fine-tuning the RoBERTa model (Liu et al., 2019a) on the SemEval tasks, including a learning rate of 0.000025, a batch size of 4, all for 16 epochs. The initial weights are based on standards followed by BERT and RoBERTa (Devlin et al., 2019; Liu et al., 2019a). The remaining parameters are based on open source implementation by HuggingFace<sup>3</sup> (Wolf et al., 2020).

Whilst the initial approach of selecting sensible defaults for hyperparameters achieved state of the art results, the random, manual process was painful, the results sometimes unpredictable. Applying Bayesian optimization, however, in running a sweep over a range of hyperparameter values helped in the selection of hyperparameters, including epoch size, batch size, and learning rates, across the 24 layers of RoBERTa<sub>LARGE</sub> model. Our system’s implementation of the Bayesian Optimization is based on the open source wandb client by Weights & Biases<sup>4</sup> (Biewald, 2020)

### 4 Experimental Setup

In this section, we present the experimental setup for the our model and hyperparameter sweeps for the SemEval tasks.

#### 4.1 Implementation

For all experiments for the RoBERTa model (using RoBERTa<sub>BASE</sub> and RoBERTa<sub>LARGE</sub> variants), we use the PyTorch<sup>5</sup> implementation of it in the HuggingFace transformer open source library<sup>6</sup> (Wolf et al., 2020) together with the simpletransformer<sup>7</sup> (Rajapakse, 2019) wrapper library. We maintain the default weights and hyperparameters whilst only changing the learning rate, batch size and finetuning for different values of epochs. All experiments were run on V100 GPUs with 16GB memory.

#### 4.2 Data Processing

The training and evaluation data, as shown in Table 1 is based on the training, development and test data supplied for the SemEval-2021 Task 7 (Meaney et al., 2021). Train data for each Subtask is the same but the the number of annotations are

<sup>3</sup><https://github.com/huggingface/transformers>

<sup>4</sup><https://github.com/wandb/client>

<sup>5</sup><https://pytorch.org>

<sup>6</sup><https://github.com/huggingface/transformers>

<sup>7</sup><https://simpletransformers.ai>

Task	Type	Metric	Train
<b>Task 1a</b>	Classification	F1-Score	8000
<b>Task 1b</b>	Regression	RMSE	4932
<b>Task 2a</b>	Regression	RMSE	8000

Table 1: A Summary of Subtasks and dataset. Both development and test set have the same have sizes of 1000.

different, with 8000 annotations for Subtask 1a and 2a; and 4932 annotations for Subtask 1b. Data splits between training and evaluation is 80% and 20%.

The annotators for the dataset for the tasks were a diverse group of individuals from differing age group (18-70), genders, political views and income levels – their backgrounds reflecting their perceptions of jokes or humor. For each text in the dataset, annotators were asked to rank as either humorous or not, and to rate the humor level on a scale of 1 to 5.

The subjectivity level of each text is also captured as a controversy score. Each text is labeled as controversial if the variance of its humor rating is greater than the median variance of all texts. Otherwise, it is labeled as not controversial.

As a way of combining humor and offense detection in the same task, a first in SemEval tasks, annotators were asked to classify humor as either offensive or not, and, if offensive, to rate the offensiveness on a scale of 1 to 5. Non-offensive humor received a zero rating.

Overall, the SemEval task divides into four subtasks. Task 1a, a binary task, predicts if a given text should be considered humorous. The second Task 1b, a regression task, assigns a rating between 1 to 5 to text considered humours, and 0 otherwise. The third Task 1c, itself a binary task, gives a controversy score to a text. The fourth task predicts the general offensiveness of a text on a scale of 0 to 5. Our system’s implementation only experiments with three of the Subtasks, including Task 1a, Task 1b, and Task 2a.

### 4.3 Hyperparameter tuning

Our approach to hyperparameter tuning involves two steps – one manual (implemented during the evaluation phase) , the other using Bayesian optimization (implemented during post-evaluation). In the first step, though, we experiment with a range of hyperparameter values on Task 1a, and the results applied to train our model on the various subtasks.

But during the second step, implemented during the post-evaluation phase, we implement hyperparameter sweeps on each task.

In the first step, we manually select from a range of tunable hyperparameters, with batch sizes  $\in \{4, 16\}$ , learning rates  $\in \{2e - 5, 4e - 5, 1e - 4\}$  and we fine-tune for epochs in  $\in \{6, 9, 12, 16\}$  . The remaining hyperparameters, including dropout rates, and the parameter weights, are based on the default values for RoBERTa model implementation in the HuggingFace transformer library.

Using the results of the first step as our initial defaults for batch size and learning rate, we consider a fine-grained hyperparameter sweep using the Bayesian optimization across the 24 pretrained layers of the RoBERTa<sub>LARGE</sub> model. We select a range of learning rates between 0 &  $1e - 3$  for the pretrained layers. We fine-tune for a range of 6 to 40 epochs, applying early stopping and using accuracy as the evaluation metric on the valuation set for Task 1a, and RMSE as the evaluation metric for Task 1b and Task 2a.

Runs on hyperparameter sweeps are taken on all the three Subtasks - Task 1a, 1b, and 2a. We then use the results of learning rates across the pretrained layers, together with the batch size to train our model for the selected number of epochs on each Subtask. Table 5 shows the results of each hyperparameter sweep.

## 5 Results

In this section we present the results of our SemEval tasks and our analysis for each step. We present, in the first step, our baseline method and results. We then follow up with the results obtained evaluation phase; here, we analyse the impact of the manual sweep and finetuning on the pretrained RoBERTa model on the subtasks. In the last step, and using the results of the codalab scores, we analyse the impact of hyperparameter sweeps on the scores during the post-evaluation phase.

Tasks	Metric	Scores evaluation
<b>Task 1a</b>	F1-Score	0.939
	Accuracy	0.926
<b>Task 1b</b>	RMSE	0.646
<b>Task 2a</b>	RMSE	0.506

Table 2: Official evaluation scores for each task.

Tasks	Metric	Scores
		post-evaluation
<b>Task 1a</b>	F1-Score	0.957
	Accuracy	0.947
<b>Task 1b</b>	RMSE	0.5802
<b>Task 2a</b>	RMSE	0.469

Table 3: Post-evaluation scores. These were scores generated during the post-evaluation stage after multiple hyperparameter sweeps.

## 5.1 Baseline

For our baseline method for the regression tasks, we use a very simple linear regression class by scikit-learn. For the classification task, though, we use logistic regression, also by scikit-learn, but with binarized ngram counts, a method proposed by Wang and Manning (2012). The baseline result for the classification task is 89%. RMSE baseline results for the regression tasks, for both Task 1b and 2a, are 0.54 and 0.74 respectively.

## 5.2 Official Evaluation

As the evaluation results in Table 2 shows, the F1-Score of 0.939 for **Task 1a** is very high, and that is even for manual values of learning rate, epoch and and batch size. What this shows is that using the recommended learning rate and batch size to fine-tune pretrained RoBERTa model on humor classification tasks can achieve very high results. On the the hands, the same hyperparameter values achieved average RMSE metric score for **Task 1b** and average F1 score for **Task 2a**, which suggest that our approach of using the same hyperparameter values for all subtasks is not working.

## 5.3 Post Evaluation

During the post-evaluation phase the team carried out an extensive hyperparameter finetuning with the bayesian optimization. Table 3 also shows substantial gain in the F1-score, 0.95, for **Task 1a**, RMSE scores (0.5802 and 0.469) for **Task 1b** and **Task 2a** respectively, after applying the results of the Bayes-optimized hyperparameter sweep in Figure 5 during the second step in the post-evaluation stage.

## 6 Error Analysis

In an attempt to measure our model’s predictions against the annotations by humans, we calculate the confusion matrix, comparing the predicted re-

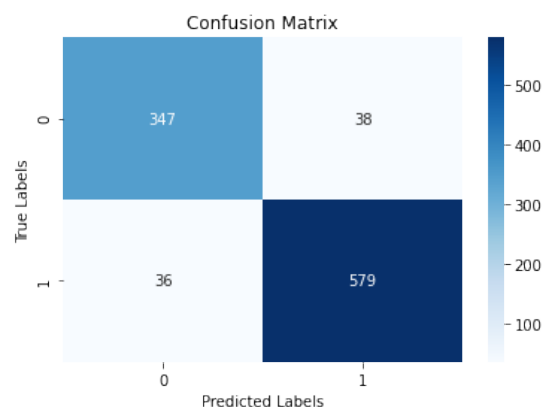


Figure 1: Confusion matrix for Subtask 1a during evaluation phase

sults with the values in the gold test for Subtask 1a. Figure 1, the confusion matrix during the evaluation phase, shows comparable results for false positives and false negatives. In Figure 2, however, the number of false positives (34) are almost twice the number of false negatives, which is so because the train set has more label 1 data (4932) than label 0 data (3068), a slight difference that can lead to the false positives. Again, the total number of true positives (596), almost twice the number of true negatives (351), shows the model is biased towards positive labels, which will make it difficult to generalize.

Moreover, as shown in Table 4, about half of the number of the false positive predictions are also offensive, in part because most of the labeled texts in the train set that are labeled as humorous are also labeled as offensive.

The higher number of accurate predictions for both the evaluation phase(926) and post-evaluation phase(947) shows our model is efficient in detecting humor.

However, the RMSE scores for humor rating – that is Task 1b, including even the improved RMSE score of 0.58 during the post evaluation phase – still lags behind the RMSE results for Task 2a , the offense-rating subtask. And what this suggest is that our model performs better with offense rating than with humor rating. And that might suggest that the higher scores associated with Task 1a are based on the model’s ability to detect offense, which explains why most of the false positive texts also contain offensive content.

ID	text	is humor	offense
9200	Black trans-masculine barbers in NY where ya at? We tryna sumn.	0	0.55
9307	What 's a Black Competitive Swim Skills Clinic? ' feeling motivated	0	1.55
9756	Being told that because I'm a Reform Jew that I'm not actually a Jew and that my smicha as a reform rabbi isn't recognized is a pretty shitty thing to hear from someone.	0	1

Table 4: Examples of gold test results wrongly labeled by our model

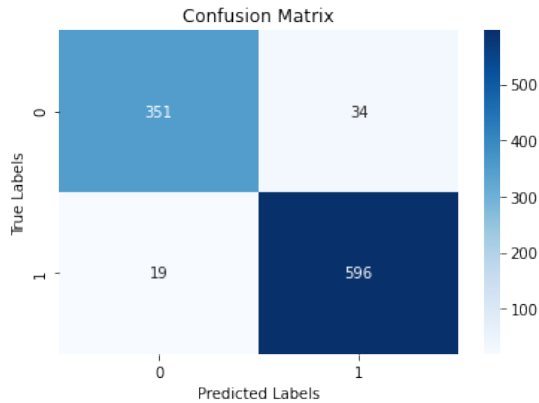


Figure 2: Confusion matrix for Subtask 1a during post-evaluation phase

## 7 Discussions and Future Works

One major limitation of our approach was that the hyperparameter runs, during the evaluation phase, were experimented only on Task 1a, a binary classification task, and the results applied on the two regression tasks to train our model, which may explain the subpar results for the Task 1b and Task 2a. However, the steps taken during the post-evaluation phase, by independently running the sweeps on each Subtask, showed substantial increase in performance.

In addition, the RoBERTa model, as implemented by HuggingFace, is used as is, without any modification to either the classification layer on top of the RoBERTa model or any of the pretrained layers. In the future, it will be worth pursuing how modifying either of the layers will impact on humor and offense detection.

Overall, however, our system shows that getting optimal values for learning rate, batch and epoch size can yield higher performance for humor detection.

## 8 Ethical Considerations

The training of RoBERTa, along with other language models such as BERT and its variants, has been shown to be costly, both for its effect on the environment and finance (Strubell et al., 2019). Again, the embeddings used for these language models tend to amplify racial, sexist, and homophobic biases. Mindful of these tendencies, our model experiments included steps to minimize bias and reduce energy cost.

What SemEval-2021 Task 7 (Meaney et al., 2021) intends to achieve is not only to rank humor but also to rate humor offensiveness, the first of any SemEval task. To achieve this, the dataset contains as much as humor as hate, which covers racial slurs, gender bias, trans/homophobic comments, etc. Knowledge of what ranks as offensive in humorous text can help our system moderate humorous content.

To ensure that dataset used for training and development do not over-represent hegemonic viewpoints, Meaney et al. (2021), organizers for the SemEval-2021 Task 7, employed annotators from disparate backgrounds, in age, gender, political views, to ensure that humor ratings and rankings, a subjective process, reflected the varied viewpoints.

Annotators were limited to English speakers, however, which implies that the system's ability to detect and identify humor is largely reflect views inherent in the English language.

Training and testing were carried out on 1 V100 GPUs with less than 16GB of memory, a step taken to ensure minimal, if any, impact on the environment.

The model, however, is prone to classifying offensive content as humorous, which may suggest that applications based on our model will be more likely to rate as humorous any content that might be deemed offensive.

## References

- James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305.
- Lukas Biewald. 2020. [Experiment tracking with weights and biases](#). Software available from wandb.com.
- Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. [Transformer-XL: Attentive language models beyond a fixed-length context](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Ian Dewancker, Michael McCourt, and Scott Clark. 2016. [Bayesian optimization for machine learning: A practical guidebook](#). *CoRR*, abs/1612.04858.
- Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. [SemEval-2020 task 7: Assessing humor in edited news headlines](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 746–758, Barcelona (online). International Committee for Computational Linguistics.
- F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *Proc. of LION-5*, page 507–523.
- Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. 2013. [An evaluation of sequential model-based optimization for expensive blackbox functions](#). In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation, GECCO '13 Companion*, page 1209–1216, New York, NY, USA. Association for Computing Machinery.
- Hairong Liu, Mingbo Ma, Liang Huang, Hao Xiong, and Zhongjun He. 2019a. [Robust neural machine translation with joint textual and phonetic embedding](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3044–3049, Florence, Italy. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. [Roberta: A robustly optimized BERT pre-training approach](#). *Computing Research Repository*, abs/1907.11692.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. [Semeval 2021 task 7, hahackathon, detecting and rating humor and offense](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Terufumi Morishita, Gaku Morio, Shota Horiguchi, Hiroaki Ozaki, and Toshinori Miyoshi. 2020. [Hitachi at SemEval-2020 task 8: Simple but effective modality ensemble for meme emotion recognition](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1126–1134, Barcelona (online). International Committee for Computational Linguistics.
- Peter Potash, Alexey Romanov, and Anna Rumshisky. 2017. [SemEval-2017 task 6: #HashtagWars: Learning a sense of humor](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 49–57, Vancouver, Canada. Association for Computational Linguistics.
- T. C. Rajapakse. 2019. [Simple transformers](#). <https://github.com/ThilinaRajapakse/simpletransformers>.
- Alon Rozenal and Dadi Biton. 2019. [Amobee at semeval-2019 tasks 5 and 6: Multiple choice CNN over contextual embedding](#). *CoRR*, abs/1904.08292.
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. [Energy and policy considerations for deep learning in NLP](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.
- Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. [SemEval-2018 task 3: Irony detection in English tweets](#). In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.
- Sida Wang and Christopher Manning. 2012. [Baselines and bigrams: Simple, good sentiment and topic classification](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea. Association for Computational Linguistics.
- Orion Weller and Kevin D. Seppi. 2019. [Humor detection: A transformer gets the last laugh](#). *CoRR*, abs/1909.00252.
- Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. [Uhh-It at semeval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection](#).

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. [Huggingface’s transformers: State-of-the-art natural language processing](#). *CoRR*, abs/1910.03771.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. [Semeval-2019 task 6: Identifying and categorizing offensive language in social media \(offenseval\)](#). *CoRR*, abs/1903.08983.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. [SemEval-2020 task 12: Multilingual offensive language identification in social media \(OffenseEval 2020\)](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

## A Appendix

The Bayesian optimization is used in hyperparameter optimization techniques for getting a combination of hyperparameters that returns the best performance as measured by a validation set. More formally, for an expensive function  $f : \mathcal{X} \rightarrow \mathbb{R}$ , hyperparameter optimization can be represented below:

$$\mathbf{x}_{opt} = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{arg\,max}} f(\mathbf{x})$$

The objective function,  $f(x)$ , represents the score to maximize – in our case, say, the accuracy score – over the validation set;  $\mathbf{x}_{opt}$  is the set of hyperparameters that will yield the highest value of the score – or, in the the case of error rate or RMSE, the lowest value.

The Bayes optimization (or search) is one of the standard algorithms for hyperparameter sweeps; the others are grid and random search (Bergstra and Bengio, 2012). Grid search creates a grid of hyper-parameters and runs through all the range of hyperparameter values. Whilst the grid search is

better than manual selection, it is computationally expensive; it is also inefficient because the choice of the next hyper-parameter values in a cycle run is not informed by previous values.

The Bayes search, on the other hand, keeps record of previous results, and uses the results to build a probabilistic model for mapping hyperparameter values to a probability of score on the objective function (Dewancker et al., 2016).

Our system implements the Sequential Model-Based Optimization (SMBO) method, which is a succinct formalization of Bayesian Optimization (Hutter et al., 2011, 2013). Sequential Model-Based Optimization technique iterates between fitting a model and then using that model to determine the next location to evaluate. The pseudocode in Algorithm 1, adopted from SigOpt<sup>8</sup> (Dewancker et al., 2016), encapsulates the technique.

---

### Algorithm 1 Sequential Model-Based Optimization

---

```

Input:  $f, \mathcal{S}, \mathcal{X}, \mathcal{M}$ 
 $\mathcal{D} \leftarrow \text{INITSAMPLES}(f, \mathcal{X})$ 
for  $i \leftarrow \mathcal{D}$  to  $T$  do
   $p(y | \mathbf{x}, \mathcal{D}) \leftarrow \text{FITMODEL}(\mathcal{M}, \mathcal{D})$ 
   $\mathbf{x}_i \leftarrow \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{arg\,max}} \mathcal{S}(\mathbf{x}, p(y | \mathbf{x}, \mathcal{D}))$ 
   $\mathbf{y}_i \leftarrow f(\mathbf{x}_i)$ 
   $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{x}_i, \mathbf{y}_i)$ 
end for

```

---

From a domain  $\mathcal{X}$  of tunable hyperparameters, the system would select a small set to initialize a probabilistic regression model  $\mathcal{M}$ . Afterwards, subsequent locations are selected sequentially from the domain by optimizing the acquisition function  $\mathcal{S}$ , which uses the current model – the Gaussian process, in our implementation – as a surrogate of the objective function  $f$ . Each suggested result is then used to produce the real result  $\mathbf{y}_i = f(\mathbf{x}_i)$ , the values of which are appended to  $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_i, \mathbf{y}_i)\}$ ; the updated values are then used by the regression during the next iteration.

<sup>8</sup><https://sigopt.com>

<b>hyperparams</b>		<b>Task 1a</b>
lr	<i>layer</i> <sub>0-5</sub>	$1.556e^{-5}$
	<i>layer</i> <sub>6-11</sub>	$4.38e^{-6}$
	<i>layer</i> <sub>12-15</sub>	$4.62e^{-5}$
	<i>layer</i> <sub>16-23</sub>	$3.24e^{-5}$
epochs		9
lr		$4.5e^{-5}$

(a) Task 1a

<b>hyperparams</b>		<b>Task 1b</b>
lr	<i>layer</i> <sub>0-5</sub>	$3.55e^{-5}$
	<i>layer</i> <sub>6-11</sub>	$9.38e^{-6}$
	<i>layer</i> <sub>12-15</sub>	$2.76e^{-5}$
	<i>layer</i> <sub>16-23</sub>	$2.79e^{-5}$
epochs		6
lr		$2.1e^{-5}$

(b) Task 1b

<b>hyperparams</b>		<b>Task 2a</b>
lr	<i>layer</i> <sub>0-5</sub>	$7.86e^{-6}$
	<i>layer</i> <sub>6-11</sub>	$1.02e^{-5}$
	<i>layer</i> <sub>12-15</sub>	$1.38e^{-5}$
	<i>layer</i> <sub>16-23</sub>	$2.08e^{-5}$
epochs		18
lr		$3.74e^{-5}$

Table 5: Hyperparameter Sweep Results for learning rates for the 24 layers of RoBERTa<sub>LARGE</sub> model, learning rate, epoch and batch sizes of the model.

# CSECU-DSG at SemEval-2021 Task 7: Detecting and Rating Humor and Offense Employing Transformers

Afrin Sultana, Nabila Ayman, and Abu Nowshed Chy

Department of Computer Science and Engineering

University of Chittagong, Chattogram-4331, Bangladesh

{afrin.sultana.cu, nabila.ayman.cu}@gmail.com,  
and nowshed@cu.ac.bd

## Abstract

With the emerging trends of using online platforms, peoples are increasingly interested in express their opinion through humorous texts. Identifying and rating humorous texts poses unique challenges to NLP due to subjective phenomena i.e. humor may vary to gender, profession, age, and classes of people. Besides, words with multiple senses, cultural domain, and pragmatic competence also need to be considered. A humorous text may be offensive to others. To address these challenges SemEval-2021 introduced a HaHackathon task focusing on detecting and rating humorous and offensive texts. This paper describes our participation in this task. We employed a stacked embedding and fine-tuned transformer models based classification and regression approach from the features from GPT-2 medium, BERT, and RoBERTa transformer models. Besides, we utilized the fine-tuned BERT and RoBERTa models to examine the performances. Our method achieved competitive performances in this task.

*Keywords:* humor and offense rating, transformers, BERT, RoBERTa, stacked embedding.

## 1 Introduction

The exponential evolution of technologies and social platforms increases the growth of user-generated content. Therefore, detecting specific information from a pile of web data is ubiquitous. Humor, like most figurative language, poses interesting linguistic challenges to NLP, due to its emphasis on multiple word senses, cultural knowledge, and pragmatic competence. The automated humor detection and rating is one of the challenging and promising tasks for their significance in the field of opinion mining, sentiment analysis, and emotion intelligence domain.

Numerous works have been done on humorous text identification task. (Weller and Seppi, 2019) proposed pre-trained BERT architecture to identify humorous jokes from Reddit dataset, puns, and short jokes dataset. (Khatri and Pranav, 2020) used BERT and GloVe embeddings with linear support vector classifier (SVC), naive Bayes, and random forest for predicting the final label to detect sarcasm in tweets. (Badlani et al., 2019) extracted features pertaining to sarcasm, humor, hate speech, as well as sentiment and ensemble them for sentiment classification. In (Liang et al., 2020), the pre-trained BERT is adapted to the distantly supervised NER (named entity recognition) task with early stopping. Some evaluation campaigns related to automatic humor detection also has been performed. (Swamy et al., 2020) includes a logistic regression baseline, a BiLSTM + attention-based learner, and a transfer learning approach with BERT to tackle the Internet humor at SemEval-2020 Task 8.

However, most of the existing work addressed the humorous text identification problem as a binary classification task. But the humorous rating of a text can be varied at different scales based on its contents. Moreover, a humorous text may portray offensive context to other users due to a variety of users' perceptions. To bridge this gap (Meaney et al., 2021) introduced a shared task at SemEval-2021 focusing on detecting and rating humorous and offensive texts. The task comprises of four subtasks: Task 1a is to predict whether a text is humorous or not, task 1b is to rate a humorous text with a score between 0 and 5, task 1c represents the subjectivity of humor appreciation by foretelling a humorous text is controversial or not, and task 2 intend to rate a text with a value between 0 and 5 based on its offensiveness.

To tackle the challenges of this task, we employed a stacked embedding of various transformer models including GPT-2 medium, BERT, and

---

The first two authors have equal contributions.



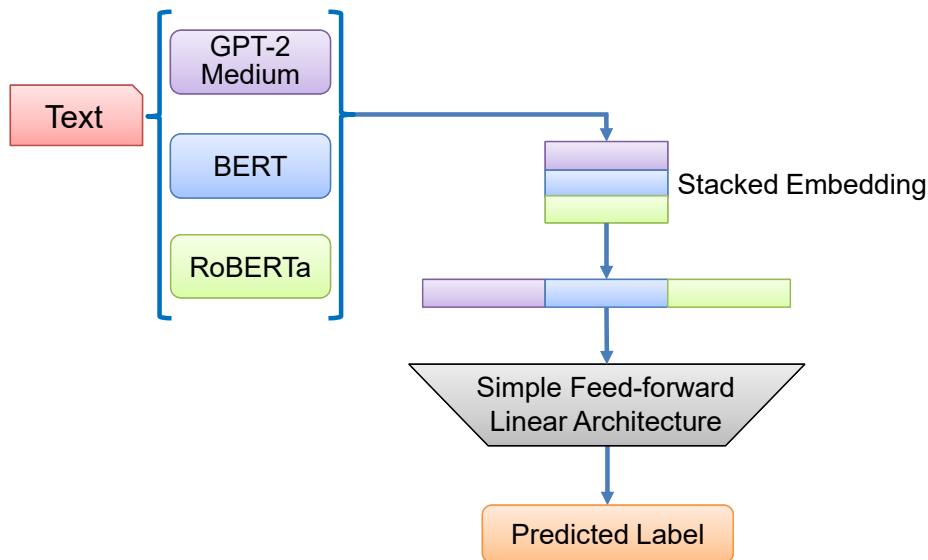


Figure 1: Proposed framework.

RoBERTa with a simple linear architecture. Besides, we conduct experiments and examine the individual performance of these three models. We used transformer-based models as they can learn the context of the sentence effectively and pushed the state-of-the-art for a wide range of downstream NLP tasks.

The rest of the paper is structured as follows: Section 2 describes our proposed framework. Section 3 illustrates our experiment and evaluation. Finally, we come to end with some conclusion and future research directions in Section 4.

## 2 Proposed Framework

In this section, we present the details of our proposed approach for humorous and offensive text identification and rating. The overview of our proposed framework is depicted in Figure 1.

Given a text, we have extracted embedding features from three state-of-the-art transformer-based models including GPT-2 medium, BERT, and RoBERTa. The extracted embeddings are then unified through the stacked embedding scheme and the unified feature vector is then passed to the simple feed-forward linear architecture to obtain the prediction score.

### 2.1 Text Encoding

**GPT-2:** GPT-2 (Radford et al., 2019) stands for generative pretrained transformer 2, which is trained on eight million text documents scraped from the web. It has an outstanding ability to gener-

ate coherent text from minimal prompts. We utilize GPT-2 medium version to get a 1024 dimensional feature vector from each text for sub-task 1a.

**BERT:** BERT (Devlin et al., 2019) stands for bidirectional encoder representations from transformers, which is a new method of pre-training sentence representations. It is trained on a large corpus of unlabelled text which includes the entire Wikipedia (that’s about 2500 million words) and a book corpus (800 million words). We deploy BERT-base uncased version with fine-tuning to get a 768-dimensional feature vector for a given text. We also conduct experiments with the fine-tuned BERT-large uncased architecture to encode each text into a 1024-dimensional feature vector.

**RoBERTa:** RoBERTa (Liu et al., 2019) stands for robustly optimized BERT pre-training approach. An improvement on BERT which introduced as a robustly optimized method for pre-training NLP systems. BERT’s language masking strategy is used in RoBERTa, wherein the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. RoBERTa modifies key hyperparameters in BERT including removal of BERT’s next-sentence pre-training objective, training with much larger mini-batches, and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better performance on downstream tasks. We use the RoBERTa base model with fine-tuning to get a 768-dimensional feature vector.

## 2.2 Stacked Embeddings

Stacked embeddings are one of the most important concepts that combine different embeddings to capture the benefits of different embedding models. Stacked embeddings concatenate embedding vectors generated from various models to form the final word vectors. The representation of stacked embedding-based framework is depicted in Figure 1. We combine GPT-2 medium, BERT-base, and RoBERTa-base embeddings using Flair’s (Akbiik et al., 2018) stacked embeddings approach. For each text, we extract a 1024-dimensional transfer learning feature vector from the GPT-2 medium, a 768-dimensional feature vector from the BERT-base uncased, a 768-dimensional feature vector from the RoBERTa-base model. We combine these embeddings and get a 2560-dimensional feature vector for each text.

## 2.3 Classification Module

The transfer learning feature vector obtained from the stacked embedding approach goes through a simple feed-forward linear architecture which applies a linear transformation to the incoming feature vectors and classifies each text either as humorous or non-humorous. Besides, we employ the fine-tuned BERT and RoBERTa based classifiers.

For the regression tasks, fine-tuned BERT and RoBERTa based regression models are utilized. We implement these models from the Huggingface transformers library (Wolf et al., 2020).

# 3 Experiments and Evaluations

## 3.1 Dataset Description

SemEval-2021 task 7 organizers (Meaney et al., 2021) provided a benchmark dataset to evaluate the performance of the participant’s proposed systems. The given training set consists of 8000 texts whereas the development set contains 1000 texts and the test set contains 1000 texts. The data set contains one row for each text. Each row has a unique identifier, the text, and the values for `is_humor`, `humor_rating`, `humor_controversy`, and `offense_rating`. If the sentence is humorous the value of `is_humor` is 1 otherwise 0. The value of `humor_rating` and `offense_rating` defines the level of humor or offensiveness in the contents. The value of `humor_controversy` determines whether the humorous text is controversial or not. The value of `humor_rating` and `humor_controversy` is subject to

Parameters List	Sub-task 1a	Other sub-tasks
Epochs	4	20
Batch size	16	16
Learning rate	3e-5	2e-6
Maximum length	default	160
Patience	3	5
Optimizer	Adam	AdamW
Anneal factor	0.5	–

Table 1: Optimal value of parameters used in this work.

the value of `is_humor`. If a given text is not humorous, `humor_rating` and `humor_controversy` contain no value for that particular text. Therefore, we converted all empty cells of `humor_rating` and `humor_controversy` to label 0. We used the train data set to train our model. The development data was used for hyperparameter tuning. Finally, we used the given test dataset to evaluate our models.

## 3.2 Evaluation Measures

The organizers employed different strategies to evaluate the performance of participants’ systems. Standard evaluation measures including F1-score and accuracy were applied to estimate the performance for sub-tasks 1a and 1c. The regression subtasks i.e. 1b and 2a were evaluated by the metric root mean squared error (RMSE).

## 3.3 Experimental and Parameter Settings

We used Google Colab’s GPU for training and parameter tuning of our system. We evaluated our system’s performances through Codalab platform.

Now, we describe the value of optimal parameters that we’ve used to design our model. While evaluating for the sub-tasks, we fine-tuned our systems based on the number of epochs, batch size, learning rate, anneal factor, patience, and optimizer to obtain the improved performances. We utilize the early-stopping method to overcome the system’s overfitting on the training dataset. We interchange epochs in range [4,8,15,20] with different value of patience, batch size in range [8,16,32], learning rate in range [2e-5, 3e-5, 4e-5, 2e-6, 3e-6, 4e-6] with optimum `max_length`. While evaluating sub-task 1a, we use Flair’s (Akbiik et al., 2018) framework and for other sub-tasks, we utilize transformers from HuggingFace transformers library (Wolf et al., 2020). We note that compared to BERT, RoBERTa has slightly different hyper-

parameters. In particular, RoBERTa uses weight decay with  $\lambda = 0.1$  and no gradient clipping (Mosbach et al., 2020). Table 1 describes the summarized parameters settings used in this work. The default settings are used for the rest of the parameters. In this paper, we reported the results based on these settings. However, if the prediction for the rating sub-tasks is less than 0.35, the value is converted into 0.0 according to our result analysis on the development set.

### 3.4 Results and Analysis

We used the full training dataset for training our proposed model and the validation set for hyperparameter tuning. The comparative performance of our method based on test data against the top-ranked participants’ systems in individual sub-task are presented in Table 2 and 3, respectively.

Team (Rank)	Accuracy	F1-Score
Comparative Performance on Subtask-1a		
PALI (1st)	0.9820	0.9854
stce (2nd)	0.9750	0.9797
<b>CSECU-DSG (34th)</b>	<b>0.9380</b>	<b>0.9496</b>
mayukh (35th)	0.9330	0.9468
Avilshmam (56th)	0.816	0.8489
Comparative Performance on Subtask-1c		
PALI (1st)	0.4943	0.6302
reynier (10th)	0.4732	0.6197
<b>CSECU-DSG (34th)</b>	<b>0.5366</b>	<b>0.4423</b>
GuanZhengyi (35th)	0.5593	0.4271

Table 2: Comparative results with other participants’ systems in binary classification tasks.

We now report the best performing model for each sub-task. Stacked embeddings based method used for sub-task 1a and BERT’s large model is used for sub-task 1c. However, for the regression subtasks 1b and 2a, BERT’s large and RoBERTa’s base models are employed, respectively.

Results showed that our proposed system obtained competitive results in sub-task 1a and 1c whereas in the other sub-tasks our system lags behind. Ensembling transformer’s embedding achieved good performance for binary classification. In our system, we have tuned a few hyperparameters and used features from all intermedi-

Team (Rank)	RMSE Score ↓
Comparative Performance on Subtask-1b	
abcbpc (1st)	0.4959
mayukh (4th)	0.5257
<b>CSECU-DSG (41th)</b>	<b>0.6803</b>
Maoqin (44th)	0.7405
JAGD (47th)	0.8847
Comparative Performance on Subtask-2a	
DeepBlueAI (1st)	0.4120
MagicPai (8th)	0.4460
<b>CSECU-DSG (34th)</b>	<b>0.5395</b>
Anik (38th)	0.5800
MLXG (47th)	0.9587

Table 3: Comparative results with other participants’ systems in regression tasks.

ate layers of transformers. We didn’t apply any approach to find out effective intermediate layers combination to obtain the best performance from transformers. Though the training set contains 8000 sentences, only 4932 sentences are humorous. That’s why we get only 4932 perfect humor-rated sentences to train the system which is scanty. All these requirements limit our system performance for regression tasks 1b and 2a.

## 4 Conclusion and Future Plan

In this paper, we have tackled the problem of identifying and grading humorous texts as defined in SemEval-2021 task 7. Achieving high performance in humor and offensive text identification or ranking is hard due to its diverse contextual form. We have presented transformer-based language models including GPT-2 medium, BERT, and RoBERTa in a unified architecture using a stacked embedding scheme. Experimental results demonstrated its effectiveness for the classification tasks. However, we have seen that finetuned transformer models performed better in the regression task.

In the future, we have a plan to focus on direct inducing topic information into the transformer-based models. We also intend to explore how to tune the parameters for regression tasks in more efficient ways, which could yield better performances.

## References

- Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual String Embeddings for Sequence Labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.
- Rohan Badlani, Nishit Asnani, and Manan Rai. 2019. Disambiguating Sentiment: An Ensemble of Humour, Sarcasm, and Hate Speech Features for Sentiment Classification. *W-NUT 2019*, page 337.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL:HLT)*, pages 4171–4186.
- Akshay Khatri and P Pranav. 2020. Sarcasm Detection in Tweets with BERT and GloVe Embeddings. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 56–60.
- Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: BERT-Assisted Open-Domain Named Entity Recognition with Distant Supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1054–1064.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. SemEval 2021 Task 7, HaHackathon, Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines. *arXiv preprint arXiv:2006.04884*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.
- Steve Durairaj Swamy, Shubham Laddha, Basil Abdussalam, Debayan Datta, and Anupam Jamatia. 2020. NIT-Agartala-NLP-Team at SemEval-2020 Task 8: Building Multimodal Classifiers to Tackle Internet Humor. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1179–1189, Barcelona (online). International Committee for Computational Linguistics.
- Orion Weller and Kevin Seppi. 2019. Humor Detection: A Transformer Gets the Last Laugh. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3612–3616.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

# RedwoodNLP at SemEval-2021 Task 7: Ensembled Pretrained and Lightweight Models for Humor Detection

**Nathan A. Chi**  
De Anza College  
Cupertino, CA  
chinathan@student.deanza.edu

**Ryan A. Chi**  
Stanford University  
Stanford, CA  
ryanchi@stanford.edu

## Abstract

An understanding of humor is an essential component of human-facing NLP systems. In this paper, we investigate several methods for detecting humor in short statements as part of Semeval-2021 Shared Task 7. For Task 1a, we apply an ensemble of fine-tuned pre-trained language models; for Tasks 1b, 1c, and 2a, we investigate various tree-based and linear machine learning models. Our final system achieves an F1-score of 0.9571 (ranked 24 / 58) on Task 1a, an RMSE of 0.5580 (ranked 18 / 50) on Task 1b, an F1-score of 0.5024 (ranked 26 / 36) on Task 1c, and an RMSE of 0.7229 (ranked 45 / 48) on Task 2a.

## 1 Introduction

Humor detection is the process of identifying sequences of text that are amusing—an important task, as such sequences are present in most channels of communication. Although humor detection comes naturally to humans, it is difficult for artificial systems to do the same. Part of the challenge is that it is debatable what constitutes humor; what one reader finds funny may be found utterly prosaic by the next. The problem is only complicated when demographic factors come into play; now, the element of offense is also a factor.

SemEval-2021 Shared Task 7 attempts to address some of these open problems (Meaney et al., 2021). Rather than definitively labeling text as humorous or not, Task 1a aims to determine whether the author intended for the sentence to be humorous, Task 1b predicts its humor rating by the average user (its first moment), and Task 1c attends to whether the variance of its humor ratings (its second moment) exceeds the median. Task 2a, meanwhile, considers the text’s average offensiveness score, a metric that often correlates with whether the author meant the text to be humorous and—perhaps equally importantly—affects whether the joke would be consid-

ered acceptable. Overall, training models to perform well on these tasks is of central importance to developing systems that are responsive to a wide range of input, whether in complete jest or meant to be taken at face value.

## 2 Dataset

We train and validate our models on the SemEval-2021 Task 7 training set (Table 6). Each English sentence is annotated for the following four labels, with continuous annotations labeled using a Likert scale from 1 to 5.

#	Description	Label
1a	Is the intention of this text to be humorous?	Binary
1b	How generally humorous is the text for the average user?	Continuous
1c	If the sentence is humorous, is the the humor controversial? <sup>1</sup>	Binary
2a	How generally offensive is the text?	Continuous

Table 1: Annotations/subtasks with their descriptions. We submit separate models for each of these tasks.

### 2.1 Train-test split

The dataset has a total of 10000 examples, split 8000–1000–1000 between train, validation, and test sets. However, the official development set lacked labels until the last phase of the competition, so we created our own held-out validation set for our experiments. Consequently, our train set has 6,400 examples, and our validation set has 1,600 examples. In our paper, all “validation set” performance is reported on this internal held-out set.

<sup>1</sup>In gold standard labels, an example is deemed controversial if its variance exceeded the median variance of all examples.

### 3 Methods

#### 3.1 Task 1a: Humor Prediction

The goal of this task is to model whether a given text is intended to be humorous. Hypothesizing that pretrained language models could effectively model the presence of humor in statements, we investigate the following models:

- **BERT** (Devlin et al., 2019) is a pretrained masked language model. We use BERT-Large in our experiments (335M parameters).
- **RoBERTa** (Liu et al., 2019) is a robustly optimized BERT pre-training approach that utilizes changes including a larger pre-training dataset and a dynamic masking pattern strategy. We use RoBERTa-Large (335 parameters).
- **ELECTRA** (Clark et al., 2020) is a pretrained model that uses a discriminative replaced-token identification loss rather than a demasking objective, resulting in greater data efficiency. We use ELECTRA-Large (336M parameters).

**Ensemble** We also investigate an ensemble which incorporates one BERT-large, one RoBERTa-large, and nine ELECTRA-large models. Our models were averaged with equal weights. Each ELECTRA model was trained with a different random seed from 100 to 900; in the row corresponding to the ELECTRA model’s performance, we have only included the result from the best seed (200).

**Pretraining details** We trained with binary cross-entropy loss for 3 epochs, using a learning rate of  $1 \times 10^{-5}$  and batch sizes of 16 (ELECTRA) and 8 (BERT, RoBERTa).

##### 3.1.1 Results

We find that all models achieve high F1 and accuracy, with ELECTRA performing the best of any individual model. However, we achieve the highest performance using our ELECTRA + BERT + RoBERTa ensemble. Notably, the ensemble achieves a slightly superior performance to each of its individual component models. Overall, we are ranked 24th out of 58 on this task, achieving an F1-score of 0.9571.

Model	# params	F1	Accuracy
BERT	335M	0.941	0.928
RoBERTa	355M	0.952	0.940
ELECTRA	336M	0.956	0.944
<i>Ensemble</i>	—	<b>0.957</b>	<b>0.946</b>

Table 2: Performance of our candidate models on the official evaluation set for Task 1a (humor prediction). Out of the individual models, ELECTRA achieves the strongest results, and ensembling the predictions of multiple pretrained models slightly helps both F1 and accuracy.

#### 3.2 Tasks 1b, 1c, and 2a: General Humor, Controversy, and Offensiveness

##### 3.2.1 Models

Despite their success on Task 1a, we were unable to achieve strong results with pretrained language models on the other tasks. Consequently, we experimented with several other machine learning methods, using lightweight features as inputs. We examine a number of different supervised learning algorithms, implemented using the **Scikit-learn** (Pedregosa et al., 2011) framework:

- **Support Vector Machine** is a lightweight classification algorithm that employs a hyperplane that divides a dataset into two subsets.
- **Random Forest** is a supervised learning technique that utilizes independently trained decision trees that sample from a random selection of data.
- **Gradient Boosting** is a technique that ensembles a number of weak learners (typically decision trees) and optimizes based on a differentiable loss function.
- **LightGBM** (Ke et al., 2017) is a highly efficient gradient boosting decision tree that takes advantage of GOSS (gradient-based one side sampling) and EFB (exclusive feature bundling).
- **AdaBoost** (Schapire, 1999) (**Adaptive Boosting**) is an instance of gradient boosting that optimizes by re-weighting weak learners based on high-weight data points (rather than using a differentiable loss function).
- **Multilayer Perceptron** is a feed-forward deep neural network.

Model	Features	F1 (1c)	Accuracy (1c)	RMSE (1b)	RMSE (2)
AdaBoost	GloVe	0.48	0.48	0.564	1.355
CatBoost	GloVe	0.51	0.51	0.563	0.877
GradientBoosting	GloVe	0.52	0.52	0.572	0.848
<b>LGBM</b>	<b>GloVe</b>	0.50	0.50	0.552	<b>0.808</b>
Logistic Regression	GloVe	0.52	0.52	—	—
Logistic Regression	Manual	0.50	0.53	—	—
MLP	GloVe	0.49	0.49	0.562	0.798
<b>RandomForest</b>	<b>GloVe</b>	0.52	0.53	<b>0.548</b>	0.928
<b>SVM</b>	<b>GloVe</b>	<b>0.55</b>	<b>0.55</b>	0.551	0.874
XGBoost	GloVe	0.52	0.52	0.556	0.858

Table 3: Validation set performance of candidate models on Task 1b, 1c, and 2a (controversy classification). For tasks 1b (humor rating), 1c (humor controversy), and 2a (offense rating), the highest-performing models are the random forest model with  $n_{trees} = 1000$ , the support vector machine, and the LGBM, respectively. We did not run experiments for entries marked —.

- **CatBoost** (Dorogush et al., 2018) is a variant of gradient boosting that prioritizes low latency via symmetric trees.
- **XGBoost** (eXtreme Gradient Boosting) (Chen and Guestrin, 2016) is an implementation of gradient boosting that efficiently makes use of parallel computation.

### 3.2.2 Features

Given that the subjectivity of humor is often correlated with the subject matter of the joke, we also examine its impact on humor controversy in an alternative approach to Task 1c. Often, a joke regarding a sensitive topic may be comical to one reviewer but downright unamusing to a second, whose sense of humor is entirely disparate from the first’s.

In this approach, we use a suite of engineered one-hot features with logistic regression (Table 4). Our manual features consist of groups that are typically stereotyped: more specifically, each manual feature consists of a set of tokens, and its value is the number of times a token from its set appears in the input.

In an effort to interpret the significance of these features, we calculate logistic regression (LR) coefficients with respect to the controversy label. The results show that several features were unrelated or inversely correlated to humor controversy (most notably the “Black” feature); they also indicate that a few were strongly positively correlated (such as the “White” feature).

For our final models, we use 300-dimensional GloVe word vectors (Pennington et al., 2014) mean-pooled over each sentence.

### 3.2.3 Results

The official evaluation set performances for our Transformer-based models in Task 1a are listed in Table 2, while the unofficial validation set performances for our regressors and classifiers are listed in Table 3.

For Task 1b (humor rating), we achieve the highest performance using our Random Forest model. Overall, we are ranked 18th out of 50 on this task, achieving an RMSE of 0.5580.

For Task 1c (humor controversy), we achieve the highest performance using our SVM model. Overall, we are ranked 26th of 36 on this task, achieving an F1-score of 0.5024.

For Task 2a (offense rating), we achieve the highest performance using our LGBM model. Overall, we are ranked 45th out of 48 on this task, achieving an RMSE of 0.7229.

## 4 Conclusion

We have presented models trained to predict various aspects of humor in text: the level of intended humor, the level of humor for average users, and the level of controversy and offense of a given humorous statement.

We find that large pretrained models such as ELECTRA, RoBERTa, and BERT are effective at predicting the level of intended humor. Furthermore, we note that ensembling these models slightly improves performance. However, our ex-

Feature	Description	LR Coefficient
BLACK	Words referring to those of African descent.	-0.508
AMERICAN	The word “American.”	-0.493
GENDER	Words associated with women.	-0.234
INTELLIGENCE	Words associated with stupidity.	-0.169
ISLAM	Words referring to the religion or associated institutions.	-0.102
RELIGION	All major religions not including Islam.	-0.096
RACIAL	Words referring to those of Asian, Latin American, and African descent.	-0.062
SEXUALITY	Words relating to sexuality.	-0.051
HOUSING	The word “homeless.”	-0.008
BRUTALITY	Words heavily connoting violence.	0.064
COUNTRIES	Words relating to nationalities not included in “Racial” or “American” features.	0.077
BLONDE	The word “blonde.”	0.112
PARTNER	Significant others or family members; controversial jokes often include words regarding female partners.	0.164
SEXUAL	Words relating to sexual activity.	0.171
VULGAR	Profanity.	0.242
WHITE	Words referring to those of Caucasian descent.	0.547

Table 4: Manual features for Task 1c (controversy classification)

periments highlight that pretrained models yield weaker results when faced with regression tasks, as well as when faced with the goal of trying to predict whether a given statement’s humor rating has high controversy. This may be due to difficulty in predicting inter-rater disagreement (i.e. if the humor metric’s variance exceeds the median variance).

Next, we note also that our engineered one-hot feature approach toward humor subjectivity does not perform significantly better than the baseline models. While our results do reveal a positive correlation between certain manual features and humor controversy—illustrating that humor subjectivity is to some degree affected by subject matter—our results suggest that on the whole, the effects of this relationship are limited.

Overall, our results suggest that reasonably lightweight models can achieve strong results in modelling humor in human language.

## Acknowledgments

The authors would like to thank Ethan A. Chi for his support and guidance throughout this project. We would also like to acknowledge Google Colab- oratory for their free compute services.

## References

- Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.
- Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*.
- Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis,



- Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7, hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Robert E. Schapire. 1999. A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'99*, page 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

## A Reproducibility

We release our code at <https://github.com/nathanchi/hahackathon>. Additionally, we include our hyperparameters in Table 5 for reproducibility.

Model	Hyperparameter	Task 1b	Task 1c	Task 2a
AdaBoost	learning_rate	1.0	1.0	1.0
	loss	linear	linear	linear
	$n_{\text{estimators}}$	1500	1500	1500
GradientBoosting	learning_rate	0.1	0.1	0.1
	max_depth	3	3	3
	$n_{\text{estimators}}$	1000	100	500
LGBM	learning_rate	0.1	0.1	0.1
	max_depth	10	10	10
	num_leaves	22	22	22
	$n_{\text{estimators}}$	60	600	600
MLP	learning_rate	constant	constant	constant
	$\alpha$	0.01	0.1	0.01
	$\beta_1$	0.9	0.9	0.9
	$\beta_2$	0.999	0.999	0.999
	hidden_layer_sizes	(100, 100)	(500, 500)	(200, 200)
	max_iter	12	200	12
RandomForest	$n_{\text{estimators}}$	1000	100	100
	criterion	mse	gini	mse
	max_depth	2	2	2
SVM	$C$	1.0	1.0	1.0
	degree	3	3	3
XGBoost	$n_{\text{estimators}}$	100	100	100

Table 5: Hyperparameters for lightweight supervised learning models.

Sentence	is_humor	humor_rating	humor_controversy	offense_rating
When I was in college I used to live on a houseboat and started dating the girl next door. Eventually we drifted apart.	1	2.95	0	0.25
Want to know why he disappeared? These are the most common reasons men disappear from your life.	0	0	0	0

Table 6: Examples that are intended and not intended to be humorous, respectively.

# EndTimes at SemEval-2021 Task 7: Detecting and Rating Humor and Offense with BERT and Ensembles

Chandan Kumar Pandey and Chirag Singh and Karan Mangla  
Samsung R&D Bangalore  
{chandan.p, c.singh, karan.mangla}@samsung.com

## Abstract

This paper describes Humor-BERT, a set of BERT (Devlin et al., 2019) Large based models that we used to solve the SemEval-2021 Task 7: Detecting and Rating Humor and Offense (Meaney et al., 2021). It presents pre and post processing techniques, variable threshold learning, meta learning and Ensemble approach to solve various sub-tasks that were part of the challenge. We also present a comparative analysis of various models we tried. Our method was ranked 4<sup>th</sup> in Humor Controversy Detection, 8<sup>th</sup> in Humor Detection, 19<sup>th</sup> in Average Offense Score prediction and 40<sup>th</sup> in Average Humor Score prediction globally. F1 score obtained for Humor classification was 0.9655 and for Controversy detection it was 0.6261. Our user name on the leader board is ThisIsTheEnd and team name is EndTimes.

## 1 Introduction

The purpose of this paper is to present different approaches that we tried towards various sub-tasks in SemEval-2021 Task 7 Detecting and Rating Humor and Offense task (Meaney et al., 2021). It consists of following sub-tasks:

- Task 1a: Classifying text based on whether they are humorous or not.
- Task 1b: Predicting Humor rating score
- Task 1c: If the text is classed as humorous, predict if the humor rating would be considered controversial, i.e. the variance of the rating between annotators is higher than the median
- Task 2a: predict how generally offensive a text is for users. This score was calculated regardless of whether the text is classed as humorous or offensive overall.

We participated in all of the above sub-tasks. Task 1a and Task 1c are classification problems with F1 score as metric whereas, Task 1b and Task 2a are regression problems with RMS error values as scoring criteria. All the above tasks shared the same training, development and test set (Meaney et al., 2021). Training set consisted of 8000 text sentences, containing labels for all the subtasks. Size of development and test set was 1000 text sentences each.

Humor, is an interesting linguistic challenge. It's an abstract concept and depends a lot on audience and their interpretation of the jargon used to generate humor. Notion of humor changes from culture to culture, age group, gender and social status of target audience. It is a subjective and personal phenomenon. What's humorous to one person could be non humorous or even offensive to the other. In this task, labels and ratings were collected from a balanced set of age groups from 18-70 and variety of genders, political stances and social status (Meaney et al., 2021). Humor detection and rating tasks have been the defined before but the task of detecting Controversy (Meaney et al., 2021) is new and interesting. It captures the variance in humor rating due to variance in age, gender and other demographic features of the user.

This task is also unique in the way that it combines humor and offense rating tasks for the same dataset. What is humorous to one user could be offensive to other depending on their demographic characteristics.

## 2 Related Works

Humor and offense related tasks have been pursued before as well. Various NLP techniques from classical N-gram techniques (Taylor and Mazlack, 2004) to transfer learning over pre-trained language models like BERT (Devlin et al., 2019) have been tried

for humor related tasks. (Yang et al., 2015) extract humor anchors and use a k-NN based classifier to detect humor. (Chen and Soo, 2018) used CNN and highway network whereas, (Weller and Seppi, 2019) used BERT fine tuning to classify humor in text. Apart from text, multimodal data for humor was curated by (Hasan et al., 2019). In other works on multimodal data (Yang et al., 2019) tag video comments data automatically and use audio data alone to predict humor. They used Random Forest and CNN over MFCC features. (Chen and Lee, 2017) take transcripts from TED talk and detect audience laughter using CNN.

### 3 System overview

In development phase, we split the data as training:validation :: 7200:800, but for evaluation phase we take entire 8000 texts for training and 1000 size development set. Later on, we also tried bagging, which will be described later.

#### 3.1 Models

Humor is an abstract linguistic concept, hence a model which can understand the nuances of language should be great for detecting and rating humor, offense and controversy in text. Naturally we tried one of the best language model, BERT and modified it according to the task at hand. We describe below models for each sub tasks separately.

##### 3.1.1 Task 1a: Humor Detection

Various models that we tried for this sub-task are following:

1. BERT-Large fine tune [BERT-L]
2. Fully connected layer over BERT-Large [BERT-FFN]
3. BERT-Large textual entailment [BERT-ENT]
4. CNN over BERT embedding [BERT-CNN]
5. BERT-Large based Ensemble [BERT-ENS]

For BERT-Large fine tune model(BERT-L), we used uncased BERT Large model and fine tuned over train set. We selected the best performing model over development data. For this we selected COLA as task type as it is GLUE task which models standard binary classification. It simply uses softmax classifier over CLS token from last layer of BERT Large model.

In the second approach (i.e. for BERT-FFN), we

tried a 128 size fully connected layer over CLS token from last layer and then two class softmax classifier to get humor probability scores. We also used dropout after CLS and fully connected layer. We also tried textual entailment kind of classification model over BERT Large (BERT-ENT), for this task. Each of the text data is modified to indicate whether humor is implied from any given text. For example training example:

*I'm the Michael Jordan of lazy sports analogies.# 1*  
 becomes  
*I'm the Michael Jordan of lazy sports analogies.### It is humorous. # 1*

Where label 1 represents the presence of humor in original sentence whereas, in case of textual entailment instance it represents whether "It is humorous" is implied by the text sentence. # is just a separator for illustration here. We also used entailment model to augment the dataset by adding various paraphrases of the sentence "It is humorous". For example, the original text above can be augmented in the following way:

*I'm the Michael Jordan of lazy sports analogies.### It is humorous. # 1*  
*I'm the Michael Jordan of lazy sports analogies.### It is funny. # 1*  
*I'm the Michael Jordan of lazy sports analogies.### It is not humorous. # 0*

In another approach (i.e. BERT-CNN), we tried freezing the BERT parameters and used it only to get the sentence embedding from the last layer. We took word embeddings for each word in the sentence from the last layer of BERT and then applied a one layer 1-D CNN over those embeddings, then a fully connected layer and softmax layer to detect humor.

BERT-Large based Ensemble model (i.e. BERT-ENS), is our best model which gave F-1 score of 0.9655. First thing we tried is Bagging of BERT-large models. The base model used in bagging was the model from approach 2. We split the train data in 10 random datasets (i.e. bags) of size 7200(train) and 800(validation). Train 10 base models over each of those datasets and select the best performing model on validation set of size 800. Now com-

bine these 10 models using soft/hard voting ensembling approach. We predict the labels using best models from each of the bag and then take a majority vote among the 10 selected models. We tried variants of this approach where each of the 10 models had different hyper-parameter values, for example different values of dropout and maximum sentence length varies between 64 to 128. In another approach, we sum the softmax score of each epoch of all the 10 bags and then use argmax to predict the labels for each bag. After that, we take a majority vote among 10 such models created (This was our best performing model). The development data of size 1000 was used for hyper-parameter tuning and selection of the final ensemble model. We used **Binary cross entropy** as loss function.

### 3.1.2 Task 1b: Humor Rating

Taking hint from Task 1a, we tried fewer models that we felt would perform better for this sub-task. Following models were tried

1. Fully connected layer over BERT-Large [**BERT-FFN**]
2. BERT-Large based Ensemble [**BERT-ENS**]

The architecture of models 1 is same as the corresponding models in Task 1a (3.1.1), except for the last layer and loss function. Here instead of two class softmax layer we used a linear regression layer to predict the humor rating and used **Mean squared Error** and **Root Mean squared Error** as metric to be minimized. For model in approach 3, we use the same setting as described in Task 1a (3.1.1), except that instead of voting method we used averaging of humor rating predicted by all the base 10 models. As in Task 1a, models from each bag are selected based on which model has least RMS error i.e. the best model among all epochs. In other approach, We took the average of rating predictions from all the epochs that were trained for 10 bags (This was our best model). We also tried bagging models with different hyper-parameter values as in Task 1a. The best hyper-parameter values are described in 4.3.

### 3.1.3 Task 1c: Humor Controversy

We tried models similar to the Task 1a (3.1.1) as described below.

1. BERT-Large fine tune [**BERT-L**]
2. Fully connected layer over BERT-Large [**BERT-FFN**]

### 3. BERT-Large based Ensemble [**BERT-ENS**]

The architecture of models 1 and 2 are same as the corresponding models in Task 1a (3.1.1), except for additional meta learning of softmax thresholds to predict whether the humor is controversial or not. There is class imbalance here and so models were less confident in detecting controversy in humor. Also, it's a challenging task, in the sense that detecting controversy is not so obvious. In order to balance the odds we tried various softmax score threshold values to predict controversy. Instead of taking argmax we tried different values of softmax probability score for Controversy class. For, standard binary classification a threshold of 0.5 is used to predict a particular class. We tried different lower values of threshold in favour of Controversy class, since model were not very confident in detecting it. Threshold value 0.1 worked best for approach 1 and 2.

In 3<sup>rd</sup> approach we use bagging ensemble setting similar to Task 1a (3.1.1) with the difference that we used softmax score averaging of models rather than a voting, and thereafter we used threshold tuning. Threshold value 0.15 worked best for our method. All the threshold values were obtained by fine tuning over development dataset of size 1000.

### 3.1.4 Task 2a: Average Offensiveness Score

This is a regression task, hence we tried models similar to that of Task 1b (3.1.2).

1. Fully connected layer over BERT-Large [**BERT-FFN**]
2. BERT-Large based Ensemble [**BERT-ENS**]

The architecture, loss function and all other settings are same as that of corresponding models in Task 1b (3.1.2).

## 4 Experiments

In this section, we describe the dataset and various experiments performed. For classification tasks we used **Binary Cross-Entropy** loss, whereas for regression tasks we used **Mean Squared Error** as metric to be minimized.

### 4.1 Dataset

We used the dataset provided by the SemEval-2021 Task 7 organizers (Meaney et al., 2021). Dataset consisted of 8000 English text sentences for training and 1000 text sentences for development and

evaluation each. We trained our model on the training dataset only, no external dataset other than that was used. Dataset split for each model has already been described in section 3.

## 4.2 Pre-processing

We removed various special characters which do not contain any useful information. We also expanded emoji symbols to their meaningful definition using existing preprocessing package spaCy since they are really important for tasks such as humor detection and humor rating prediction. Afterwards, texts were lower cased and tokenized using Sentencepiece (Kudo and Richardson, 2018) tokenizer.

## 4.3 Hyperparameters

We tried various values for the hyper-parameters. The one that worked best for us is described in table 1.

Hyper-parameter	Value
Batch Size	32
Learning rate	2e-5
Maximum Sentence length	64, 128
CLS layer Dropout	0.3
FFN layer Dropout	0.5
Number of Bags	10
Number of Epochs	25
FFN Activation	<i>tanh</i>
FFN hidden size	128

Table 1: Hyper-parameter values

Changing Maximum sentence length didn't make much difference in results. Activation function *tanh* worked better than *relu* and *gelu*. CLS layer dropout is dropout after last layer of BERT. FFN layer dropout is applied after Feed forward network layer of hidden size 128. Dropout was required because the model was too complex for 8000 size dataset. No of Bags is the no of bags/no of base models we trained during Bagging ensemble approach. All the models were developed using Tensorflow (Abadi et al., 2015) library. Training was done on NVIDIA Tesla P-40 GPUs.

## 5 Results and Analysis

We describe the results for each of the sub-tasks and analyse the results. Baseline for classification

task was Naive Bayes model with bag of words features, and for the regression task, it was Support Vector Regression.

Model	F1 Score
Baseline	0.884
BERT-L	0.926
BERT-FFN	0.935
BERT-ENT	0.912
BERT-CNN	0.937
<b>BERT-ENS</b>	<b>0.966</b>

Table 2: Humor Detection Results

Model	RMS Error
Baseline	0.861
BERT-FFN	0.667
<b>BERT-ENS</b>	<b>0.654</b>

Table 3: Average Humor Score Results

Model	F1 Score
Baseline	0.462
BERT-L	0.567
BERT-FFN	0.586
<b>BERT-ENS</b>	<b>0.626</b>

Table 4: Humor Controversy Results

Model	RMS Error
Baseline	0.642
BERT-FFN	0.522
<b>BERT-ENS</b>	<b>0.469</b>

Table 5: Average Offensiveness Score Results

We see in table 2 that applying a Feed Forward network with dropout layer improves the result. This is expected since a FFN layer after BERT results in non-linear combination of BERT features. Dropout is required because model is more complex now. BERT-CNN also performs reasonably well because of efficacy of CNN in learning and combining  $N - Gram$  features. Not to mention, here we have frozen the BERT layer so no of parameters is less. Relatively poor performance of BERT-ENT can be attributed to the fact that this task is not suitable for Textual Entailment

classification. **BERT-ENS** outperforms other models by huge margin. Here we see the effect of bagging as a way to reduce error due to variance. Different base models learn different features and peculiarities. BERT-Large is a complex and strong classifier, so understandably, it has high variance. Combining multiple BERT-Large models using voting mechanism provided quite a significant jump in F1 score.

For Average Humor score results in table 3 we again see that a single BERT model with FFN does perform well but bagging reduces the error caused by high variance. We see same pattern for Average Offensiveness score. In fact the effect of bagging is even more prominent there.

Meta-learning in form of variable softmax threshold worked really great for Humor Controversy detection. Humor Controversy is even more abstract than Humor and hence difficult to detect. There is class imbalance as well, since, very few examples are actually Controversial. So, we had to lower the softmax probability threshold values for Humor Controversy class. Individual BERT model is a weak learner in this case that's why combining them results in a huge jump in F1 score. On comparing our results with the baseline models we see that even a single BERT-Large model outperforms the baseline by large margin. This solidifies our notion that a great language model like BERT will always be a top performer in NLP tasks. And, combining the BERT with ensemble methods has potential to outperform other competing models.

## 6 Conclusion and Future Work

Humor is a abstract linguistic construct. That is why a strong language model like BERT-Large performs really well on these tasks. Our models were ranked under 10 in 2 tasks and under 20 in 1 task, this shows that BERT based models outperform other models. Especially, if many BERTs are combined using a good ensemble technique. For future work ensemble methods like stacking and blending could be tried. Also, different models with different hyper parameter values could be combined in a more effective way to get better results.

## References

- Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. [TensorFlow: Large-scale machine learning on heterogeneous systems](#). Software available from tensorflow.org.
- Lei Chen and Chong Min Lee. 2017. [Predicting audience's laughter during presentations using convolutional neural network](#). In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 86–90, Copenhagen, Denmark. Association for Computational Linguistics.
- Peng-Yu Chen and Von-Wun Soo. 2018. [Humor recognition using deep learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117, New Orleans, Louisiana. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Md Kamrul Hasan, Wasifur Rahman, AmirAli Bagher Zadeh, Jianyuan Zhong, Md Iftekhar Tanveer, Louis-Philippe Morency, and Mohammed (Ehsan) Hoque. 2019. [UR-FUNNY: A multimodal language dataset for understanding humor](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2046–2056, Hong Kong, China. Association for Computational Linguistics.
- Taku Kudo and John Richardson. 2018. [SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and*

*the 11th International Joint Conference on Natural Language Processing.*

Julia M Taylor and Lawrence J Mazlack. 2004. Computationally recognizing wordplay in jokes. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 26.

Orion Weller and Kevin Seppi. 2019. [Humor detection: A transformer gets the last laugh](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3621–3625, Hong Kong, China. Association for Computational Linguistics.

Diyi Yang, Alon Lavie, Chris Dyer, and Eduard Hovy. 2015. [Humor recognition and humor anchor extraction](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376, Lisbon, Portugal. Association for Computational Linguistics.

Zixiaofan Yang, Bingyan Hu, and Julia Hirschberg. 2019. Predicting humor by learning from time-aligned comments. In *INTERSPEECH*, pages 496–500.



# IITH at SemEval-2021 Task 7: Leveraging transformer-based humorous and offensive text detection architectures using lexical and hurtlex features and task adaptive pretraining

Tathagata Raha, Ishan Sanjeev Upadhyay, Radhika Mamidi, Vasudeva Varma

IIT Hyderabad, India,

{tathagata.raha, ishan.sanjeev}@research.iiit.ac.in

{radhika.mamidi, vv}@iiit.ac.in

## Abstract

This paper describes our approach (IITH) for SemEval-2021 Task 5: HaHackathon: Detecting and Rating Humor and Offense. Our results focus on two major objectives: (i) Effect of task adaptive pretraining on the performance of transformer based models (ii) How does lexical and hurtlex features help in quantifying humour and offense. In this paper, we provide a detailed description of our approach along with comparisons mentioned above.

## 1 Introduction

Humour is an important part of human conversation. It has a social function as well and can play an important role in group cohesiveness(Ziv, 2010). Hence humorous content is also found on various social media websites. While there has always been a fine line between funny and offensive humour, the anonymity, distance and isolation provided by being online can increase instances of offensive or controversial humour being posted online. (Weitz, 2017)

In this task, we have presented a transformer based approach combined with lexical and hurtlex feature sets to quantify humour and offense of a piece of text.

We achieved an F1 score of 0.959 in the humor classification task and 0.592 in the humor controversy task. For the regression tasks, we achieved a RMSE score of 0.541 and 0.488 in the humor regression and offense regression task respectively.

## 2 Related work

There have been many attempts made at computational humour detection. In this section, we briefly describe other work in this area. In this approach(Blinov et al., 2019), the authors have used universal language model fine-tuning method

for humour recognition. Convolutional neural networks (CNN) have also been used for this task by (Chen and Soo, 2018) whereas (Weller and Seppi, 2019) used transformers to classify humour.

There has also been a lot of shared tasks and workshops related to computational humour. One of them is SemEval-2020 Task 7: Assessing Humor in Edited News Headlines(Hossain et al., 2020) where Zhang(Zhang et al., 2020) used bidirectional neural networks with an attention mechanism and incorporated lexical features to assess humour in edited news headlines.

There has been a lot of work done on hate speech and offensive speech detection as well. CNN's and gated recurrent units (GRU) have been used for this task (Zhang and Luo, 2018). Recurrent neural networks combined with user-related information have also been used for hate speech detection in Twitter Data (Pitsilis et al., 2018) whereas multilingual transformer architectures were leveraged by (Ghosh Roy et al., 2021) to detect hostile content in English, Hindi and German.

## 3 Task and dataset overview

The task(Meaney et al., 2021) is divided into 4 sub-tasks.

1. **Humour detection:** This is a binary classification task where the model needs to predict if the text is humorous or not where the values are either 0 and 1.
2. **Humour Rating:** This is a regression task where the model needs to rate how humorous the text is where the value can vary between 0 to 5.
3. **Controversy detection:** This is a binary classification task where the model needs to classify text as controversial or not if it has been classified as humorous. It can be either 0 or 1.

4. **Offense Rating:** This is a regression task where the model needs to rate how offensive the text is. It can vary between 0 to 5.

The dataset for the tasks was provided by The workshop organizers. It consisted of 10,000 sentences. 8,000 sentences were provided for training and 1,000 for validation. The remaining 1,000 were used for testing. Each row consisted of a unique identifier, the text and the label values of "is\_humor", "humor\_rating", "humor\_controversy" and "offense\_rating".

## 4 Methodology

### 4.1 Hurltlex features

HurtLex (Bassignana et al., 2018) is a lexicon of offensive, aggressive, and hateful words in over 50 languages which is further categorized into 17 categories. Identifying these kinds of words can potentially help in offensive content detection. Also, in some cases, a humorous piece of text might contain such a word to denote humour. We have also experimented with this feature for humour classification and regression task.

### 4.2 Lexical features

The structure of humorous and offensive texts can be a bit different from normal texts. We have leveraged a lexical feature set that would help us capture that information and distinguish humorous and offensive texts. The set of lexical features are:

- Counting the total number of letters, punctuation, upper case letters and numbers within the text.
- Identifying the presence of any named entity. For detecting named entities, we have used the AllenNLP named entity recogniser<sup>1</sup> which uses pretrained GloVe vectors for token embeddings and a GRU encoder. (Peters et al., 2017)
- Detecting the presence of interrogation by identifying "??" symbol or any WH-word
- Detecting the number of personal pronouns and what kind of personal pronouns they are: first-person, second-person or third-person.

<sup>1</sup><https://demo.allennlp.org/named-entity-recognition/named-entity-recognition>

For detecting the personal pronouns, we have used a pre-defined list of personal pronouns.

### 4.3 Sentence embeddings

For generating the sentence embeddings, we have experimented with 4 different pre-trained transformer models: bert-base-uncased (Devlin et al., 2018), roberta-base (Liu et al., 2019), google/electra-base-discriminator (Clark et al., 2020) and xlnet-base-cased (Yang et al., 2019). Initially, we finetuned each of the pre-trained models for each task and made predictions on the validation set. On the basis of the performance, we have selected one pre-trained model to proceed to our final setup 4.5. For the binary humour classification, humour regression and offensive regression task, we have selected roberta-base. On the other hand, google/electra-base-discriminator gave the best performance for humour controversy task.

### 4.4 Task adaptive pretraining

In the paper (Gururangan et al., 2020), we can see the benefits of continued pretraining of pre-trained transformer models on unlabelled task-specific data or Task Adaptive Pretraining (TAPT) before finetuning them on a downstream task like text classification. This paper (Raha et al., 2021) showcases the gains attributed to further pre-training of the IndicBERT (Kakwani et al., 2020) model for hostility detection in Hindi. We have experimented with the same approach for all our downstream tasks where a pretrained transformer model (roberta-base for humor classification, regression and offensive regression) is further pretrained on training data with the masked language modelling (MLM) objective. In our results 5, we have shown the benefits gained from task adaptive pretraining for each task. Note that task adaptive pretraining was not done on google/electra-base-discriminator for the humour controversy classification.

### 4.5 Final setup

In this subsection, we outline our final architecture from the set of input features to the final label generation for each task.

At first, we have generated the set of lexical features and the hurltlex features on both training, validation and testing data. For generating the hurltlex features, we have used the featurizer in hurltlex

Setting	Task 1a (F1-Score)	Task 1b (RMSE)	Task 1c (F1-Score)	Task 2 (RMSE)
TRANS	0.944	0.572	<b>0.592</b>	0.522
TRANS + LEX	0.956	0.547	0.521	0.524
TRANS + HURT	0.949	0.570	0.347	<b>0.488</b>
TRANS + LEX + HURT	<b>0.959</b>	<b>0.541</b>	0.375	0.505

Table 1: Results on the Validation split for each task with and without hurtlex and lexical features. TRANS refer to transformer embeddings, LEX refer to lexical features and HURT refers to hurtlex features. Task 1a refers to humour classification, Task 1b refers to humour regression, Task 1c refers to humour controversy and Task 2 refers to the offensive regression task. For Task 1a, 1b and 2 we have used the TAPT roberta-base and for task 1c we have used pre-trained google/electra-base

Github repository <sup>2</sup>. We have used Pytorch(Paszke et al., 2019) <sup>3</sup> and Pytorch Lightning as our primary deep-learning framework <sup>4</sup>. For our pre-trained transformer models, we chose the roberta-base <sup>5</sup> and google/electra-base-discriminator<sup>6</sup> as a part of HuggingFace’s Transformers library. For performing the task adaptive pretraining(TAPT) on downstream tasks, we have used AllenAI’s implementation of Task Adaptive Pretraining<sup>7</sup>. The roberta-base model was further pretrained on MLM objective for 100 epochs with the other hyperparameters being set to their default values. For all the transformer architectures, we have set the maximum sequence length to 128. As this is a classification task, we have used the embeddings of [CLS] as the transformer representation of the whole sentence.

Finally, the embeddings generated from the transformer models are concatenated with hurtlex features and lexical features to form the final vector representation for a particular text. For optimization, we have used the Adam (Kingma and Ba, 2017) optimizer where the learning rate was set to 1e-5 and a dropout (Srivastava et al., 2014) with the probability of 0.1. We updated weights based on cross-entropy loss values for the classification tasks and Mean Squared Error for the regression tasks. A dense multi-layer perceptron serves as the final binary classifier head or regression head. The model weights were saved and evaluated on the development set at the end of every epoch and the finetuning continued for 10 epochs. We have

<sup>2</sup><https://github.com/valeriobasile/hurtlex>

<sup>3</sup>[pytorch.org/](https://pytorch.org/)

<sup>4</sup><https://www.pytorchlightning.ai/>

<sup>5</sup><https://huggingface.co/roberta-base>

<sup>6</sup><https://huggingface.co/google/electra-base-discriminator>

<sup>7</sup>[github.com/allenai/dont-stop-pretraining](https://github.com/allenai/dont-stop-pretraining)

Task	Without TAPT	With TAPT	Gains
Task 1a (F1-Score)	0.933	0.944	0.011
Task 1b (RMSE)	0.616	0.572	0.044
Task 2 (RMSE)	0.579	0.522	0.057

Table 2: Results on the Validation split for each task with and without Task Adaptive Pretraining(without considering the lexical and hurtlex features). Task 1a refers to humour classification. Task 1b refers to humour regression and Task 2 refers to the offensive regression task.

reported the scores of the models that yielded the best F1 score on the development set and used them to further predict on the test set. We have also experimented with or without considering the hurtlex and lexical features to showcase the gains or losses attributed to them.

## 5 Results

The gains attributed to task adaptive pretraining of roberta-base on the humour classification is shown in table 2. We can see that continued pretraining of roberta-base has improved the model performances significantly.

In table 1, we can see the results of inclusion and exclusion of the lexical and hurtlex features for each task. We notice that lexical and hurtlex features do contribute to the performance of humour classification. Combining hurtlex features and lexical features with transformer embeddings have improved the results of both humour classification and humour regression task. For offensive regression, the hurtlex features played an important role while lexical features degraded the performance. This is probably because the lexical features were curated for the identification of humour. For the

Task	Score	Rank
Task 1a (F1-Score)	0.9616	14
Task 1b (RMSE)	0.5263	5
Task 1c (F1-Score)	0.6242	6
Task 2 (RMSE)	0.4772	23

Table 3: Results on the test split for each task and their respective ranks on the leaderboard during the evaluation phase. Task 1a refers to humour classification, Task 1b refers to humour regression, task 1c refers to humor controversy and Task 2 refers to the offensive regression task.

humour controversy, excluding lexical and hurtlex features gave the best results. This might be because textual features played much more important role than lexical and hurtlex features.

In table 3, we report the results obtained on the test set during the evaluation phase and the rank of our models on the official leaderboard<sup>8</sup>. We used the best performing models on the validation set to achieve those results.

Overall, this work shows how task adaptive pre-training can improve model performance for downstream tasks and the role of hurtlex and lexical features for humor and offensive detection.

## 6 Conclusion

All the experiments performed above were done with default hyperparameters (unless explicitly mentioned) due to resource constraints. The model performances could have improved if we could search for optimal hyperparameters using cross validation. Furthermore, the regression tasks could improve if we could use an ensemble of the best performing models for our final predictions.

## References

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Vladislav Blinov, Valeria Bolotova-Baranova, and Pavel Braslavski. 2019. [Large dataset and language model fun-tuning for humor recognition](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4027–4032, Florence, Italy. Association for Computational Linguistics.

<sup>8</sup>[http://smash.inf.ed.ac.uk/tasks\\_results/hahackathon\\_results.html](http://smash.inf.ed.ac.uk/tasks_results/hahackathon_results.html)

Peng-Yu Chen and Von-Wun Soo. 2018. [Humor recognition using deep learning](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 113–117, New Orleans, Louisiana. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sayar Ghosh Roy, Ujwal Narayan, Tathagata Raha, Zubair Abid, and Vasudeva Varma. 2021. Leveraging multilingual transformers for hate speech detection. *arXiv e-prints*, pages arXiv–2101.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Nabil Hossain, John Krumm, Michael Gamon, and Henry Kautz. 2020. Semeval-2020 task 7: Assessing humor in edited news headlines. *arXiv preprint arXiv:2008.00304*.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. [IndicNLP Suite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. [Adam: A method for stochastic optimization](#).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

J.A. Meaney, Steven R. Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. 2021. Semeval 2021 task 7: Hahackathon, detecting and rating humor and offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani,

- Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [Pytorch: An imperative style, high-performance deep learning library](#).
- Matthew E. Peters, Waleed Ammar, Chandra Bhagavathula, and R. Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.
- Georgios K. Pitsilis, Heri Ramampiaro, and Helge Langseth. 2018. [Effective hate-speech detection in twitter data using recurrent neural networks](#). *Applied Intelligence*, 48(12):4730–4742.
- Tathagata Raha, Sayar Ghosh Roy, Ujwal Narayan, Zubair Abid, and Vasudeva Varma. 2021. Task adaptive pretraining of transformers for hostility detection. *arXiv preprint arXiv:2101.03382*.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. [Dropout: A simple way to prevent neural networks from overfitting](#). *Journal of Machine Learning Research*, 15(56):1929–1958.
- Eric Weitz. 2017. [Editorial: Humour and social media](#). *The European Journal of Humour Research*, 4:1.
- Orion Weller and Kevin Seppi. 2019. [Humor detection: A transformer gets the last laugh](#).
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Tiantian Zhang, Zhixuan Chen, and Man Lan. 2020. [ECNU at SemEval-2020 task 7: Assessing humor in edited news headlines using BiLSTM with attention](#). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 995–1000, Barcelona (online). International Committee for Computational Linguistics.
- Ziqi Zhang and Lei Luo. 2018. [Hate speech detection: A solved problem? the challenging case of long tail on twitter](#).
- Avner Ziv. 2010. The social function of humor in interpersonal relationships. *Society*, 47:11–18.

# FII FUNNY at SemEval-2021 Task 7: HaHackathon: Detecting and rating Humor and Offense

**Mihai Samson**

Alexandru Ioan Cuza University Iasi  
mihai.samson@info.uaic.ro

**Daniela Gifu**

Alexandru Ioan Cuza University Iasi  
daniela.gifu@info.uaic.ro

## Abstract

The “HaHackathon: Detecting and Rating Humor and Offense” task at the SemEval 2021 competition focuses on detecting and rating the humor level in sentences, as well as the level of offensiveness contained in these texts with humoristic tones. In this paper we present an approach based on recent Deep Learning techniques by both trying to train the models based on the dataset solely and by trying to fine tune pretrained models on gigantic corpus.

## 1 Introduction

The figurative language of Social Media is one of the most challenging topics facing natural language processing (NLP). In this study, we refer at humor that requires a multidisciplinary approach for its detection (Dan Alexandru and Daniela Gifu, 2020). Imagine, a viral topic on social media as elections (Gifu, 2010) or a political crisis (Delmonte, Rodolfo and Tripodi, Rocco and Gifu, Daniela, 2013). Social media users themselves introduce a specific language based on common practices (e.g., humor, irony), making their message analysis very challenging (Reyes, Antonio and Rosso, Paolo and Buscaldi, Davide, 2012). The legitimate research questions of this paper intend to answer: Is humor an insurmountable barrier for Artificial Intelligence (AI)? We propose an approach based on recent DL techniques for sentiment analysis (SA). Furthermore, we experimented with multiple types of DL architectures ranging from Convolutional Neural Networks (CNN) to Recurrent Neural Networks (RNN) which we tried to train using only the dataset provided by the SemEval-2021 Task 7 competition, but we also used pretrained Transformer architectures which we fine-tuned using the available data. The rest of the paper is organized as follows: section 2 describes the literature related to sentiment analysis and humor detection, section

3 presents the dataset and method of this study, section 4 summarizes the results of the conducted experiments, with discussions after experiments, followed by section 5 with the conclusions.

## 2 Related Work

This topic has attracted significant attention in recent years, evidenced by increasing number of workshops of the same competition (e.g., SemEval-2017 Task 6: HashtagWars: Learning a Sense of Humor or SemEval-2020 Task 7: Assessing Humor in Edited News Headlines). Such a competition is attractive, especially since the problem of labeled data is somewhat solved, considering the fact that the automatic humor recognition depends on these. For the binary task, as in this case, there are many computational models to solve it or to detect the humor intensity or humor dimension (Yang, Diyi and Lavie, Alon and Dyer, Chris and Hovy, Edward, 2015) (Chen, Peng-Yu and Soo, Von-Wun, 2018). Thus, work on this topic was never followed by high results, as this problem is still almost subjective and text classification even for humans is very controversial and biased. Never the less, the task can be approached like any SA task for which most of the authors used LSTMs (Murthy, Dr and Allu, Shanmukha and Andhavarapu, Bhargavi and Bagadi, Mounika, 2020) or CNNs (Ouyang, Xi and Zhou, Pan and Li, Cheng Hua and Liu, Lijun, 2015). New approaches concentrate on using attention based methods (Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Lukasz and Polosukhin, Illia, 2017), in particular transformer architectures such as BERT (Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, 2018), RoBERTa (Liu, Yinhan and Ott, Myle and Goyal, Naman and Du, Jingfei and Joshi, Mandar and Chen, Danqi and

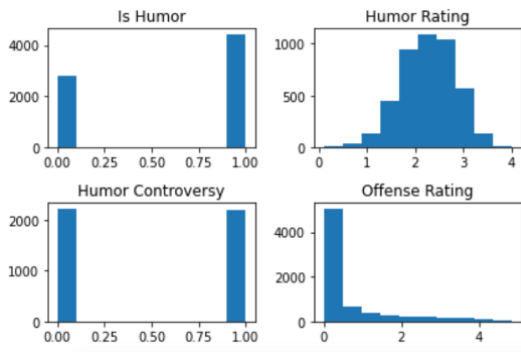


Figure 1: Data distributions by subtasks.

Levy, Omer and Lewis, Mike and Zettlemoyer, Luke and Stoyanov, Veselin, 2019), ALBERT (Lan, Zhenzhong and Chen, Mingda and Goodman, Sebastian and Gimpel, Kevin and Sharma, Piyush and Soricut, Radu, 2019), and VideoBERT (Sun, Chen and Myers, Austin and Vondrick, Carl and Murphy, Kevin and Schmid, Cordelia, 2019). These transformers are pre-trained on unlabeled data to be later fine-tuned for a variety of tasks. For this task we used the BERT architecture.

### 3 Dataset and Methods

This section contains details about the dataset built as part of SemEval-2021 Task 7 HaHackathon: Detecting and Rating Humor and Offense, which was the basis for solving the subtasks of this competition.

#### 3.1 Dataset

The dataset (Meaney, J.A., and Wilson, Steven R. and Chiruzzo, Luis and Lopez, Adam and Magdy, Walid, 2021) consists of 8000 short texts that have four labels corresponding to the four subtasks of the competition. The first label is binary and determines if the text is humorous. If this label is 1, then it has associated two additional labels: one for the second task which is a number from 0 to 5 representing the average humorous score of the annotators and another denoting if the kind of humor is controversy, again a binary classification. Regardless of the previous 3 scores, the fourth is score from 0 to 5 denoting if the text is offensive. Because of these conditions we used the entire dataset only for the first and the fourth tasks, and for the second and third we only used the texts which were labeled to be humorous. Table 1 shows some examples from the dataset of SemEval-2021 Task 7.

Note that for the third task, the labels seemed to be randomly assigned, neither of our methods succeeding in obtaining a better performance than one we would obtain by flipping a coin. Because of this situation we will only present the results for the remaining three subtasks.

#### 3.2 Method

In order to apply DL-based modeling techniques, we first need to embed the words in a vector form that the neural networks can work with. In order to achieve this, we used two methods depending on the architecture trained. For the models that we trained from scratch we used a tokenizer implemented in the Keras library <sup>1</sup> (TextVectorization) which performs the following steps exemplified on the sentence: “The quick brown fox jumps over the lazy dog.”

1. lowercasing and punctuation stripping; “the quick brown fox jumps over the lazy dog”
2. splitting the text into words; [“the”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”]
3. assembling tokens, assessing each token an index; “the”:1, “quick”:2, “brown”:3, “fox”:4, “jumps”:5, “over”:6, “lazy”:7, “dog”:8
4. transforming the text into a sequence of integers. [1, 2, 3, 4, 5, 6, 1, 7, 8]

After the tokenization, we mapped these indexes to the words from the GloVe embeddings which contains a vocabulary size of 400k words, more than enough for our task. We chose the predefined embedding size of 100 and standardized the texts to a length of 70. Figure 2 shows a histogram of the number of words after splitting the texts by space.

We also fine-tuned a pretrained BERT model which uses its own set of embeddings. In this case the huggingface <sup>2</sup> library which also provides the pretrained model offers the tools to embed the words into the necessary vectors specific to the selected model. After having the embeddings for either set of methods, we further describe the architectures used. We trained the following models from scratch:

<sup>1</sup><https://keras.io/>

<sup>2</sup><https://huggingface.co/transformers/>

id	text	is humor	humor rating	controversy	offense rating
35	Learn from the scars of others	0			0.05
119	What do you call a sad terrorist? A crisis	1	2.16	1	0.85
81	January is the Monday of months	1	2.43	0	0.00

Table 1: Examples of texts and their humor labels.

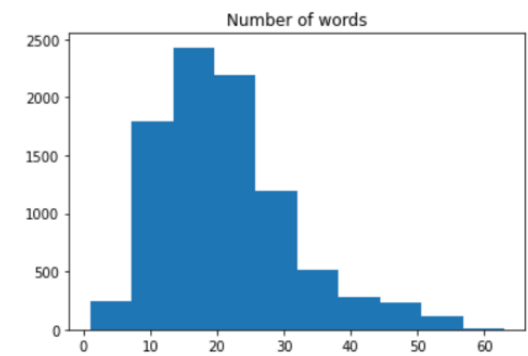


Figure 2: Histogram for the number of words in texts.

1. Convolutional Neural Networks (CNN). We created simple sequential models with 1D convolutions and max pooling layers followed by global average pooling and a few fully-connected layers. We haven't experimented with more advanced architectures like ResNet (Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, 2015) and Inception (Christian Szegedy and Wei Liu and Yangqing Jia and Pierre Sermanet and Scott E. Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich, 2014) because we found that even the simplest model drastically overfits the data.
2. Bidirectional LSTM and GRU cells. In order to take advantage of the natural structure of sentences we used the two most popular recurrent cells and we also wrapped them into a Bidirectional wrapper. Each cell was used in a separate architecture and we didn't stack more than one cell because we again found that it tends to overfit.
3. Transformer blocks. In order to take advantage of the recent developments in the NLP tasks, we created the encoder part of the original Attention is all you need paper. We used six encoding blocks followed by a global average pooling layer and a few fully-connected layers.

We employed a pretrained transformer model (BERT) from the huggingface library which was trained on cased datasets. We used the base model which has around 100 million parameters and used our dataset to only fine tune this model. Because only the first and fourth tasks used the entire dataset, we applied all these methods only on the first task, the classification between humorous and non-humorous. As it can be observed from Figure 1 the dataset is imbalanced, therefore we applied class-weight on the 0 class such that the loss from the two classes among the entire dataset will be equal. After experimenting with the hyperparameters (mainly with the dropout rate which finally we chose it to be 0.2) on this single task, the best configuration was used on the other three tasks, changing only the activation function for the regression tasks from Sigmoid to Rectified Linear Units (ReLU). For the classification tasks, the threshold for the prediction was 0.5 and for the regression ones the predictions from the neural networks were sealed to 5 in order to correspond to the competition's requirements. A diagram summarizing the system architecture can be seen in Figure 3.

The first step is to split randomly the available dataset into a training set and a validation set. We kept 10% of data for the validation set. Then, the next step is to preprocess the data. In order to have a good representation of the metrics on the validation set, we only computed the necessary tools for preprocessing on the training set and then applied these tools to the validation step. Most importantly, the vocabulary used in the models was selected based on the training set solely. For the pretrained BERT model, the tokenizer already contains a large vocabulary meant to cover most of the common words. After preprocessing the dataset, we moved forward to train the models with the corresponding embeddings. As mentioned, we did all the experiments on the first task and we experimented with the hyperparameters on this set. We did not employ a test set due to the small size of the dataset. Finally, we adapted the classification model trained



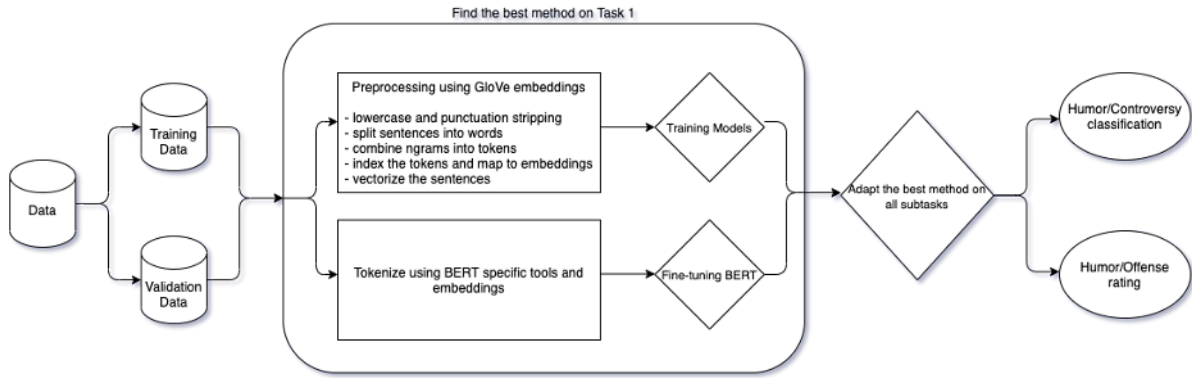


Figure 3: System Architecture.

Task Name	RMSE	Position
Average Humor	0.5598	19 of 50
Average Offensiveness	0.4788	24 of 48

Table 2: Official scores for the regression tasks.

previously to the regression tasks and trained a separate model for each task using only the samples that had the corresponding labels. After all the four models have been trained, we made predictions on the evaluation set using the tokenizer for BERT and the corresponding model.

#### 4 Results and Discussion

Below are presented the official results for all subtasks. For the classification tasks we also report the results in the post-evaluation phase and the ranking as of March 2021. In the official competition we accidentally performed a mistake for the Humor Detection task and reverted the labels that we submitted. That is why we obtained such a poor performance and this is the reason we also report these results as it better reflect the actual performance. We report Accuracy (Acc), F1-score (F1), and Root Mean Square Error (RMSE). The official results are summarized in Tables 2 and 3 and the results in the post-evaluation phase are presented in Table 4.

The results on our validation set for all the techniques are summarized in the graphics depicting the evolution of the metrics over the epochs (Figure 4). The test label refers to our validation set and not the official test set from the competition.

We can easily see that all the methods overfitted the training set. Techniques such as dropout and regularization have been applied, but we observed that they only delayed the moment when overfitting

occurred and did not increase the performance on the validation set. The best accuracy on the validation set of the techniques used on the first task is presented in Table 5.

As it can be observed from the table, the best method turned out to be to fine tune a pretrained BERT model, therefore we used this technique on the rest of the tasks. Despite of its success we can still observe that the accuracy on the validation set (95.4%) differs from the one we obtained in the competition (92.2%). For approaches to SA on a small dataset, the best way is to fine tune pretrained models, rather than trying to train a model from scratch with the available data. The humor detection task turned out to be a very challenging one, fact that can be best expressed by the results on the second classification task where we assume that the very diverse interpretations of the annotators on what constitutes a controversy humor made the task impossible to solve with any DL model. But for more approachable tasks like the first classification task or the regression tasks, probably the more consensus among annotators made the task more tractable to an artificial intelligence model.

#### 5 Conclusion

According with the legitimate question, we still consider that detecting someone’s sense of humor is a difficult problem and identifying the level of offensiveness and irony is an even harder one. In this case, a figurative content could be consider irony, satire, joke, and sarcasm. As with humor, all these figures of speech depends on the listener or reader to be in on this context. This paper presents a system participating at SemEval 2021 Task 7 and tried to adapt existing Deep Learning techniques to the problem of humor detection. This approach indi-

Task Name	Acc	F-Score	Position
Humor Detection	7.8%	0.063	58 of 58
Humor Controversy	50.08%	0.4752	29 of 36

Table 3: Official scores for the classification tasks.

Task Name	Acc	F-Score	Position
Humor Detection	92.2%	0.9374	27 of 39
Humor Controversy	50.08%	0.4752	24 of 41

Table 4: Scores in the post-evaluation phase for the classification tasks as of March 2021.

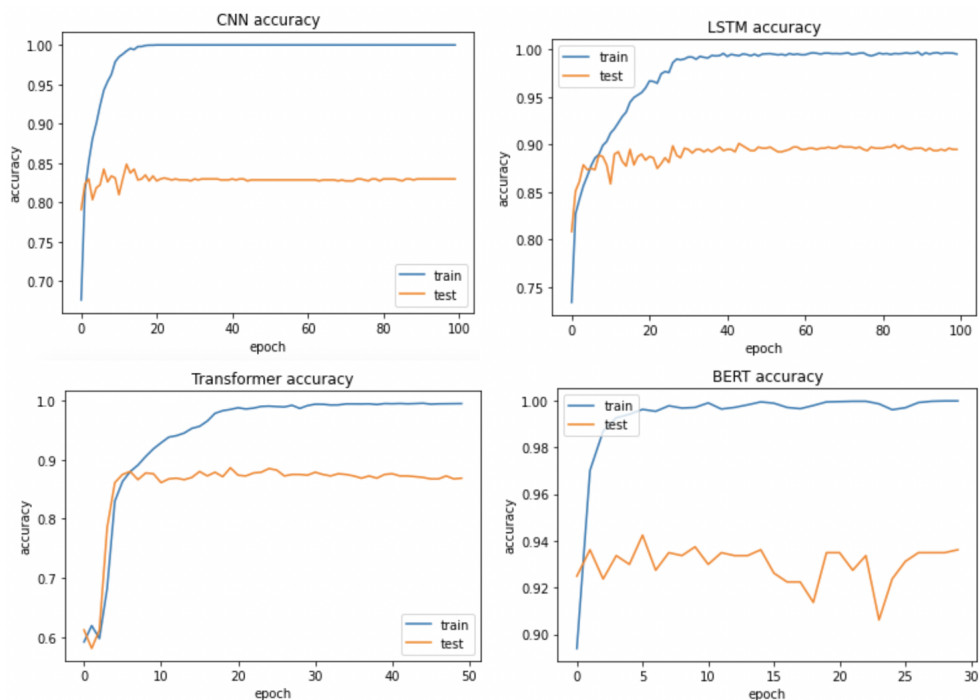


Figure 4: Evolution of accuracy .

Model	CNN	GRU	LSTM	Transformer	BERT
Validation Acc	84.9%	90.1%	90.1%	88.6%	95.4%

Table 5: The accuracy on the validation set for the 5 techniques on the Humor detection task.

cates promising results since they offer compelling results regarding the accuracy. We also found that we can get the best performance by adapting pre-trained models on other, bigger, datasets, indicating that the internal representation of language of the model acquired in other contexts can be extremely helpful in trying to identify the humor of a sentence. For further research ideas, we consider trying to use data augmentation techniques such as replacing words with synonyms, as well as using similar datasets in order to increase the dataset and the performance of the discussed methods.

## References

- Chen, Peng-Yu and Soo, Von-Wun. 2018. [Humor Recognition Using Deep Learning](#). pages 113–117.
- Christian Szegedy and Wei Liu and Yangqing Jia and Pierre Sermanet and Scott E. Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich. 2014. [Going Deeper with Convolutions](#). *CoRR*, abs/1409.4842.
- Dan Alexandru and Daniela Gîifu. 2020. [Tracing Humor in Edited News Headlines](#). In *Ludic, Co-design and Tools Supporting Smart Learning Ecosystems and Smart Education*, pages 187–196. Springer Singapore.

- Delmonte, Rodolfo and Tripodi, Rocco and Gifu, Daniela. 2013. Opinion and Factivity Analysis of Italian Political Discourse. In *IIR*, pages 88–99. Citeseer.
- Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun. 2015. [Deep Residual Learning for Image Recognition](#). *CoRR*, abs/1512.03385.
- Lan, Zhenzhong and Chen, Mingda and Goodman, Sebastian and Gimpel, Kevin and Sharma, Piyush and Soricut, Radu. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.
- Liu, Yinhan and Ott, Myle and Goyal, Naman and Du, Jingfei and Joshi, Mandar and Chen, Danqi and Levy, Omer and Lewis, Mike and Zettlemoyer, Luke and Stoyanov, Veselin. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Meaney, J.A., and Wilson, Steven R. and Chiruzzo, Luis and Lopez, Adam and Magdy, Walid. 2021. SemEval 2021 Task 7, HaHackathon, Detecting and Rating Humor and Offense. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.
- Murthy, Dr and Allu, Shanmukha and Andhavarapu, Bhargavi and Bagadi, Mounika. 2020. Text based sentiment analysis using LSTM. *Int. J. Eng. Res. Tech. Res*, 9(05).
- Ouyang, Xi and Zhou, Pan and Li, Cheng Hua and Liu, Lijun. 2015. Sentiment analysis using convolutional neural network. In *2015 IEEE international conference on computer and information technology; ubiquitous computing and communications; dependable, autonomic and secure computing; pervasive intelligence and computing*, pages 2359–2364. IEEE.
- Reyes, Antonio and Rosso, Paolo and Buscaldi, Davide. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.
- Sun, Chen and Myers, Austin and Vondrick, Carl and Murphy, Kevin and Schmid, Cordelia. 2019. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473.
- Vaswani, Ashish and Shazeer, Noam and Parmar, Niki and Uszkoreit, Jakob and Jones, Llion and Gomez, Aidan N and Kaiser, Lukasz and Polosukhin, Illia. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.
- Yang, Diyi and Lavie, Alon and Dyer, Chris and Hovy, Eduard. 2015. Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

# Counts@IITK at SemEval-2021 Task 8: SciBERT Based Entity And Semantic Relation Extraction For Scientific Data

Akash Gangwar\*    Sabhay Jain\*    Shubham Sourav\*    Ashutosh Modi

Indian Institute of Technology Kanpur (IIT Kanpur)  
{akashgnr, sabhayj, ssourav}@iitk.ac.in  
ashutoshm@cse.iitk.ac.in

## Abstract

This paper presents the system for SemEval 2021 Task 8 (MeasEval). MeasEval is a novel span extraction, classification, and relation extraction task focused on finding quantities, attributes of these quantities, and additional information, including the related measured entities, properties, and measurement contexts. Our submitted system, which placed fifth (team rank) on the leaderboard, consisted of SciBERT with [CLS] token embedding and CRF layer on top. We were also placed first in Quantity (tied) and Unit subtasks, second in MeasuredEntity, Modifier and Qualifies subtasks, and third in Qualifier subtask.

## 1 Introduction

SemEval 2021 Task 8 (Harper et al. 2021) is a task for extracting entities and semantic relations between them from a corpus of scientific articles coming from different domains. Instead of just identifying quantities, the task gives more weightage to parsing and extracting important semantic relations among the extracted entities. This is challenging because texts are ambiguous, and inconsistent, and extraction relies heavily on implicit knowledge. The results of this task can also be used for extractive scientific data summarization.

Given a scientific text, the task is to identify the span of quantities, units, and other attributes of those quantities and related measured entities, properties, and qualifiers, if any. The organizers have divided the task into five subtasks and submissions will be evaluated against all five sub-tasks<sup>1</sup>.

1. **Quantity Extraction:** For each paragraph of text, identify all Quantity spans.

\* Authors equally contributed to this work.

<sup>1</sup><https://competitions.codalab.org/competitions/25770>

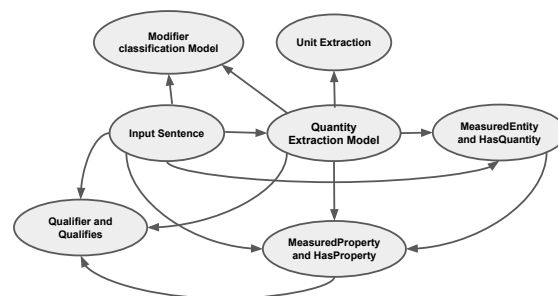


Figure 1: Overview of our proposed approach

2. **Unit Detection and Modifier Classification:** For each identified Quantity, identify the Unit of measurement and classify additional value Modifiers (count, range, approximate, mean, etc.) that apply to the Quantity.
3. **MeasuredEntity and MeasuredProperty Extraction:** For each identified Quantity, identify the MeasuredEntity and MeasuredProperty associated with it.
4. **Qualifier Extraction:** Identify and mark the span of any Qualifier that is needed to record additional related context.
5. **HasQuantity, HasProperty and Qualifies Extraction:** Identify relationships between Quantity, MeasureEntity, MeasuredProperty, and Qualifier.

We consider subtask 1 as an entity extraction task, and subtask 3, 4, and 5 are viewed as relation extraction tasks. After extracting the quantities, other attributes (MeasuredEntity, Property, and Qualifier) related to those quantities need to be predicted. The directed graph in Figure 1 gives an overview of our proposed approach. The set of incoming edges to each node represents the input to the trained model (represented by node), and the label at each node represents the prediction made by the model. The task data is extracted from CC-BY ScienceDirect Articles and made available by

the Elsevier Labs via the OA-STM-Corpus<sup>2</sup>. This motivated the use of SciBERT (Beltagy et al. 2019) model for various subtasks. “SciBERT leverages unsupervised pretraining on a large multi-domain corpus of scientific publications to improve performance on downstream scientific NLP tasks”.

Our final submitted system consisted of SciBERT with [CLS] token embedding and CRF layer on top, and it achieved an overall F1-overlap score of 0.432. We were ranked fifth on the global leaderboard. The top performance on the leaderboard achieved an overall F1-overlap of 0.519. The implementation of our system is made available via Github<sup>3</sup>.

The rest of this paper is arranged as follows. Section 2 introduces the previous work in this field and describes the organizers’ dataset. Section 3 explains our overall approach. Section 4 contains the experimental setup for training the model. We conclude with the analysis of our model performance in section 5 and concluding remarks in section 6.

## 2 Background

### 2.1 Related work

Magge et al. 2018 attempted to recognize the Clinical Entities using a LSTM CRF based architecture. The authors used the word and character level embedding obtained from word2vec (Mikolov et al. 2013). For relation extraction between these entities, authors build a binary classifier using random forest classifier. This approach has higher time complexity as it checks for all possible relationships that could exist and classifies them. The more recent work in entity extraction is by Lee et al. 2019, where they fine-tuned the BERT model using the Bio-Medical data, and have shown SOTA performance. Some other works in entity extraction includes Taher et al. 2020, where they fine-tuned BERT followed by a fully connected layer and a CRF layer.

The work by Wu and He 2019 on Relation Extraction uses BERT to identify the different types of relations between pair of entities in the given text. The system does not automatically recognize the entities between which relation exists, rather entities of interest need to be manually specified.

<sup>2</sup><https://github.com/elsevierlabs/OA-STM-Corpus>

<sup>3</sup><https://github.com/akashgnr31/Counts-And-Measurement>

### 2.2 Task setup

The scientific articles in the training and test corpus are from the following sub-domains: Astronomy, Engineering, Medicine, Materials Science, Biology, Chemistry, Agriculture, Earth Science, and Computer Science. These articles were manually annotated. The inter-annotator agreements was calculated using Krippendorff’s Alpha IAA score (Table 1).

Class	IAA Score
Quantity	0.943
MeasuredEntity	0.640
MeasuredProperty	0.545
Qualifier	0.333
Units	0.866

Table 1: IAA scores of various classes

The training dataset comprised of 298 paragraphs containing 1164 quantities, 1148 measured entities, 742 measured properties, and 309 qualifiers. The evaluation set included 135 paragraphs.

## 3 System overview

### 3.1 Pre Processing

Since we are using the SciBERT model, a maximum of 512 tokens can be passed as input to the model. Therefore, we used SciSpaCy (Neumann et al. 2019) to split the paragraph into sentences, and these sentences were passed as input to the SciBERT model.

### 3.2 Subtask 1 (Quantity Extraction)

Input sentences were tokenized using a SciBERT tokenizer from HuggingFace (Wolf et al. 2020) implementation. The Quantity span were transformed into BIO / IOB format (Ramshaw and Marcus 1995) and used as the true-labels for training the model.

The tokenized sentence is passed through SciBERT. Tanh activation function is applied over the final hidden state of SciBERT i.e.

$$H'_i = W_1[\tanh(H_i)] + b_1 \quad i = 0, 1, \dots, len$$

Here  $H_i$  is the hidden units corresponding to token  $i$  and  $len$  is the maximum length of the tokenized sentence. Similarly, [CLS] token is processed.

$$H'_{cls} = W_0[\tanh(H_0)] + b_0$$

Finally, we get the final representation for the sentence by concatenating  $H'_{cls}$  and  $H'_i$  and this is used for prediction via the softmax.

$$H''_i = W_2[\text{concat}(H'_i, H'_{cls})] + b_2 \quad i = 0, 1, \dots, len$$

$$H'' = [H''_0, H''_1, \dots, H''_{len}]^T$$

$$p = \text{softmax}(H'', \text{dim} = -1)$$

Matrices  $W_0$  and  $W_1$  have same dimension, i.e.,  $W_0 \in R^{d \times d}$ ,  $W_1 \in R^{d \times d}$ ,  $W_2 \in R^{t \times 2d}$ , where  $d$  is the hidden state size from BERT and  $t$  represent the number of tags, i.e.,  $t = 3$  in our case as we are using BIO encoding.

CRF (Conditional Random Field) (Lafferty et al. 2001) is a probabilistic model that makes it possible to extract structural dependencies among the BIO tags. The tag probability vector for all the tokens, i.e.,  $p$ , is passed through the CRF layer to generate the most probable output sequence.

We trained the model using CRF loss and Adam optimizer. The overall architecture of the model is shown in Figure 2. The tuned hyper-parameters are reported in appendix A.1.

### 3.3 Subtask 2 (Unit Detection)

The Quantity phrases are tokenized using Spacy (Honnibal et al. 2020) character-based tokenizer. The true-label for training is formatted as a binary vector marking one at the indices for characters in the unit’s span in the Quantity phrases.

We trained a Character-based Bi-LSTM (Hochreiter and Schmidhuber 1997) model with trainable word embeddings using BCE (Binary Cross Entropy) loss and Adam optimizer. The model architecture and tuned hyper-parameters are reported in appendix A.2

### 3.4 Subtask 2 (Modifier Classification)

We formulated this subtask as a multi-label classification problem with 12 labels (HasTolerance, IsApproximate, IsCount, IsList, IsMean, IsMeanHasSD, IsMeanHasTolerance, IsMeanIsRange, IsMedian, IsRange, IsRangeHasTolerance, None). To enable the BERT module to capture the location of a quantity, we insert the special symbol “\$” at the beginning and end of the Quantity span. If there are multiple Quantities in a sentence, multiple copies of the same sentence are generated with “\$” at different positions. Suppose  $H_i$  to  $H_j$  are the final hidden state vector for the Quantity span. Then, the average operation is applied to get the vector

representation of the Quantity. The averaged output is passed through a fully connected layer followed by softmax activation.

$$H'_q = W \left[ \tanh \left( \frac{1}{j-i+1} \sum_{k=i}^j H_k \right) \right] + b$$

$$p_q = \text{sigmoid}(H'_q)$$

Matrix  $W$  has dimension  $R^{l \times d}$ , where  $l$  represents the number of classification label, i.e.,  $l = 12$  in our case and  $d$  is the hidden state size from BERT.

The above model was trained using BCE (Binary Cross Entropy) and Adam optimizer. The threshold value for prediction was determined using cross-validation. The model architecture and tuned hyper-parameters are reported in appendix A.3.

### 3.5 Subtask 3 and 5 (MeasuredEntity and HasQuantity Extraction)

As done in the previous subtask to capture the location, we insert the special symbol “\$” at the beginning and end of the quantity span. The modified sentences are tokenized using a SciBERT tokenizer. The span of the MeasuredEntity related to Quantity enclosed in the “\$” symbol is transformed into BIO / IOB format and used as the true-label for training the model.

The formatted data is used to train a model similar to the Quantity Extraction (SciBERT + CRF Model). The above model extracts the MeasuredEntity associated with the Quantity enclosed in “\$”. Thus, it predicts the MeasuredEntity as well as the HasQuantity relationship of the predicted MeasuredEntity.

### 3.6 Subtask 3 and 5 (MeasuredProperty and HasProperty Extraction)

To extract MeasuredProperty and HasProperty relationship, we used a similar approach as used for MeasuredEntity and HasQuantity. We enclosed the Quantity span in “\$” symbol and the MeasuredEntity span in “#” symbol. The modified sentences are passed through the SciBERT tokenizer. The span of MeasuredProperty related to MeasuredEntity, Quantity pair is transformed into BIO / IOB format and used as the true-label for training the model.

The formatted data is used to train a model similar to the Quantity Extraction (SciBERT + CRF Model). The model trained is used to extract MeasuredProperty linked with the MeasuredEntity, Quantity pair. If the above model predicts

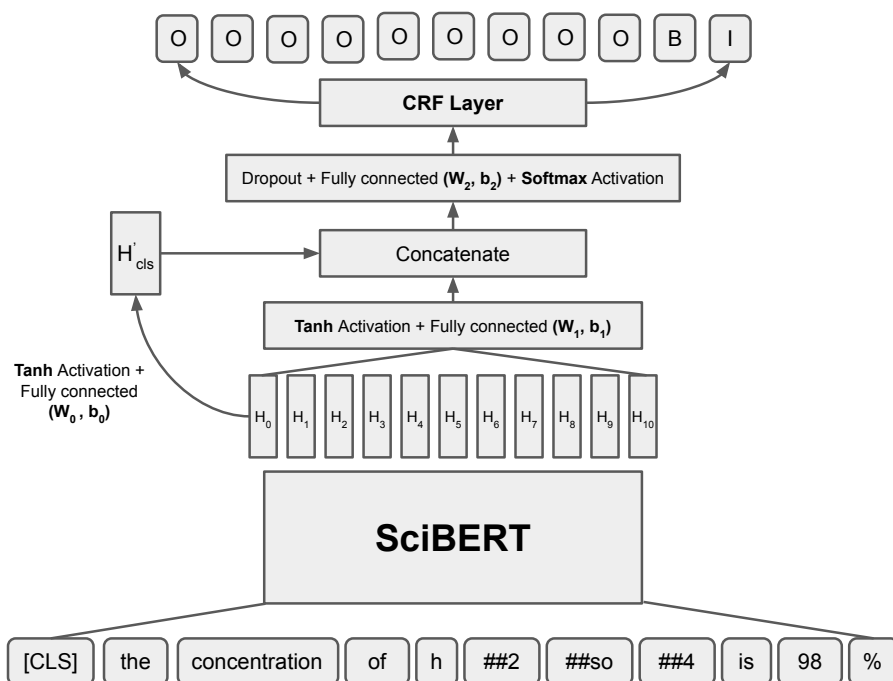


Figure 2: SciBERT with [CLS] token embedding and CRF layer on top (SciBERT + CRF Model)

Measured-Property’s span, then the HasQuantity relation is updated to MeasuredProperty, and the HasProperty relation is added to MeasuredEntity.

### 3.7 Subtask 4 and 5 (Qualifier and Qualifies Extraction)

To extract Qualifier and Qualifies’s span, two separate models similar to Quantity Extraction (SciBERT + CRF Model) were trained. While training the first model, we insert “\$” at the beginning and end of the Quantity span because we assumed that Qualifier Qualifies Quantity. During the second model training, we enclosed the MeasuredProperty span in “\$” because of the assumption that Qualifier Qualifies MeasuredProperty.

### 3.8 Post Processing

Once the predictions from all the models are available, we need to transform the predicted BIO/ IOB format into entity span format. We initially map each token’s span in the tokenized sentence and use it to determine the predicted entity’s span. While finding the span of the MeasuredEntity, MeasuredProperty, or Qualifier, if our model predicts multiple entities, then we predict the one which is closest to the Quantity span. After that, we convert the sentence span of each entity extracted to the paragraph span.

## 4 Experimental Setup

The dataset is split into two parts - train set and dev set in a ratio of 90:10. The models were trained on the train set and were validated on the dev set. The environment and packages used for training and pre-processing are listed in appendix B.

### 4.1 Evaluation Metrics

The official metrics used by the SemEval organizer are F1-measure, F1-overlap, and Exact Match. Exact Match is a binary value of 0 or 1, while F1-measure is a token level overlap ratio of submission to true spans, where tokenization is done using simple white space delimiters. F1-overlap is a SQuAD (Rajpurkar et al., 2016) style Overlap score based on F1-measure, which penalizes the negative submissions more strictly. The final evaluation is based on a global F1-overlap score averaged across all subtasks.

## 5 Results

### 5.1 Model Variants Used

We tried various models like BERT-Base, BERT-Medium (Devlin et al., 2018), SciBERT, and BioBERT (Lee et al. 2019). We could not try BERT-Large due to computational limitations. The results for the top two models are shown in Table 2.

We also experimented with Bi-LSTM layers on top of BERT, but the model was overfitting due

Model	Data Set	Quantity	Unit	Modifier	MeasuredEntity	MeasuredProperty
SciBERT	eval	0.861	0.804	0.614	0.406	0.245
SciBERT	dev	0.887	0.744	0.696	0.322	0.216
BERT-Med.	eval	0.791	0.675	0.379	0.302	0.163

Table 2.A

Model	Data Set	Qualifier	HasQuantity	HasProperty	Qualifies	Overall F1 overlap
SciBERT	eval	0.077	0.311	0.183	0.064	<b>0.432</b>
SciBERT	dev	0.083	0.270	0.137	0.083	0.410
BERT-Med.	eval	0.0	0.193	0.114	0.0	0.330

Table 2.B

Table 2: Table 2.A represents the F1-overlap score for subtask 1, 2, 3, and Table 2.B represents the F1-overlap score for subtask 4, 5 and overall F1-overlap

Metric	SciBERT + CRF	Base line
Precision	0.703	-
Recall	0.560	-
F-Measure	0.623	-
F1-overlap	0.432	0.239
Exact Match	0.371	0.211

Table 3: Overall Results on evaluation set

to its high complexity. Consequently, it was not included in the final model.

## 5.2 Results on evaluation set

The results achieved on the evaluation set for each subtask are shown in Table 2, and the overall results are shown in Table 3. Figure 3 represents the results achieved in the various subdomains.

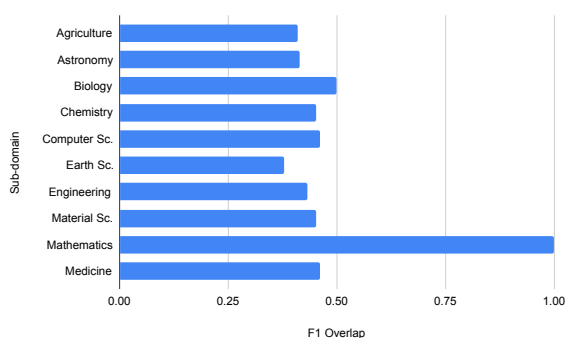


Figure 3: F1-Overlap scores of various sub-domains

The difference between the Exact Match score and F1-overlap score shows that the spans predicted by our model were precisely the same as gold data whenever they matched.

We achieved an overall fifth rank (among 19 participating teams) in the competition. We were also placed first in Quantity (tied) and Unit subtasks,

second in MeasuredEntity, Modifier and Qualifies subtasks, and third in Qualifier subtask.

## 5.3 Error Analysis

The relation extraction subtask was challenging because associating entities with the quantities they are related to is context-dependent and based on one's understanding. This is also evident from the IAA scores reported for the train data that even humans can achieve deficient performance.

Some of the aspects where our model did not work well are:

1. Our model looks for relations only within a sentence, which may cause problems when a relation exists outside the same sentence.
2. There is loss in reconstructing the TSV files from entities because the neighboring data may/maynot be part of the same entity group
3. Our model didn't work well on MeasuredProperty and Qualifiers as it did on other subtasks, which is evident as we achieved only 0.53 and 0.35 F1-overlap on training data for these two subtasks.

## 6 Conclusion

This paper proposed SciBERT + CRF Model (SciBERT with [CLS] token embedding and CRF layer on top) for span extraction, classification, and semantic relation extraction. Our model shows significant improvement in performance over the baseline model and works equally well across all the scientific sub-domains. In the future, we plan to explore various other pre-trained contextual models for our approach.



## References

- Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. [SciBERT: Pretrained Contextualized Embeddings for Scientific Text](#). *CoRR*, abs/1903.10676.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding](#). *CoRR*, abs/1810.04805.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. [Long Short-Term Memory](#). *Neural Comput.*, 9(8):1735–1780.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. [Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data](#). In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. [BioBERT: a pre-trained biomedical language representation model for biomedical text mining](#). *Bioinformatics*.
- Arjun Magge, Matthew Scotch, and Graciela Gonzalez-Hernandez. 2018. [Clinical NER and Relation Extraction using Bi-Char-LSTMs and Random Forest Classifiers](#). volume 90 of *Proceedings of Machine Learning Research*, pages 25–30. PMLR.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient Estimation of Word Representations in Vector Space](#).
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. [ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing](#). In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy. Association for Computational Linguistics.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. [SQuAD: 100,000+ Questions for Machine Comprehension of Text](#).
- Lance Ramshaw and Mitch Marcus. 1995. [Text Chunking using Transformation-Based Learning](#). In *Third Workshop on Very Large Corpora*.
- Ehsan Taher, Seyed Abbas Hoseini, and Mehrnosh Shamsfard. 2020. [Beheshti-NER: Persian Named Entity Recognition Using BERT](#). *CoRR*, abs/2003.08875.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. [Transformers: State-of-the-Art Natural Language Processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Shanchan Wu and Yifan He. 2019. [Enriching Pre-trained Language Model with Entity Information for Relation Classification](#).

## Appendix

### A Model Training

#### A.1 SciBERT + CRF

In this section we provide hyper-parameter values (Table 4) we used for training our final model to facilitate reproducibility of our results.

Hyper-parameters	Value
Hidden State Dimension ( $d$ )	768
Number of tags ( $t$ )	3
Dropout	0.1
Batch Size	24
Max Length ( $len$ )	255
Learning Rate	$10^{-5}$

Table 4: Hyper-parameters

#### A.2 Unit Detection (Character-based Bi-LSTM)

In this section we provide model architecture (Figure 4) and hyper-parameter values (Table 5) we used for training our final unit extraction model to facilitate reproducibility of our results.

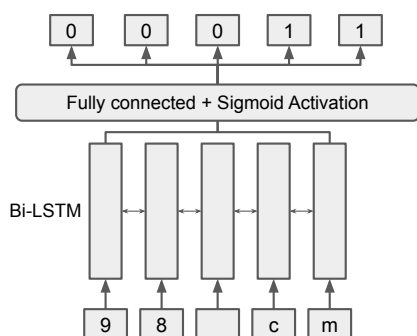


Figure 4: Character-based Bi-LSTM

Hyper-parameters	Value
Hidden State Dimension of Bi-LSTM	32
Number of Bi-LSTM layers	1
Batch Size	38
Max Length ( $len$ )	64
Learning Rate	$10^{-4}$

Table 5: Hyper-parameters

#### A.3 Modifier Classification (SciBERT with embedding averaging)

In this section we provide model architecture (Figure 5) and hyper-parameter values (Table 6) we used for training our modifier classification final model to facilitate reproducibility of our results.

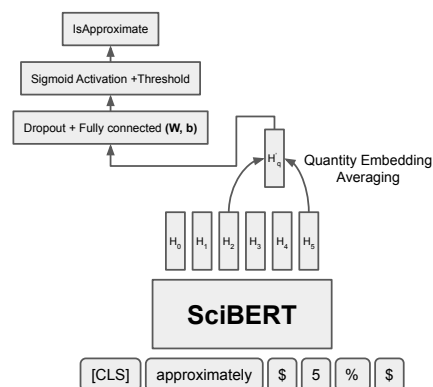


Figure 5: SciBERT with embedding averaging

Hyper-parameters	Value
Hidden State Dimension ( $d$ )	768
Number of labels ( $l$ )	12
Dropout	0.1
Batch Size	24
Max Length ( $len$ )	255
Learning Rate	$10^{-5}$
Threshold	0.5

Table 6: Hyper-parameters

### B Tools/Libraries used

We used Google Colab Nvidia T4 GPU (16GB) for training purpose. Python packages (along with version) used for pre-processing and training are tabulated below:

Package	Version
transformers	4.3.2
torchcrf	0.7.2
torch	1.7.0
scikit-learn	0.22.2
en_core_sci_sm	0.3.0
Stanza	1.2
spaCy	2.3.5
NLTK	3.2.5
pandas	1.1.5
NumPy	1.19.5

Table 7: Python Packages

# Jarvis Lab at SemEval-MeasEval Task 8: A Cascade Count and Measurement Extraction Tool for Scientific Discourse

Jiarun Cao, Yuejia Xiang, Yunyan Zhang, Zhiyuan Qi, Xi Chen\*, Yefeng Zheng

Jarvis Lab, Tencent

{jerryrcjrcao, yuejiaxiang, yunyanzhang, jasonxchen, yefengzheng}@tencent.com, qizhyuan@gmail.com

## Abstract

This paper presents our contribution to *SemEval 2021 Task 8: MeasEval*. The purpose of this task is identifying the counts and measurements from clinical scientific discourse, including quantities, entities, properties, qualifiers, units, modifiers, and their mutual relations. This task can be induced to a joint entity and relation extraction problem. Accordingly, we propose CONNER, a cascade **count and measurement** extraction tool that can identify entities and the corresponding relations in a two-step pipeline model. We provide a detailed description of the proposed model hereinafter. Furthermore, the impact of the essential modules and our in-process technical schemes are also investigated. Our code is released and available at <https://github.com/yuejiaxiang/CONNER>.

## 1 Introduction

Clinicians are currently coping with a massive amount of information, both from raw experimental data and scientific publications recording their results. However, the ever-expanding information sources have exceeded the ability of clinicians to digest and utilize them properly (Botsis et al., 2011; Cao et al., 2018; Zhou et al., 2010). Clinical information extraction tools (Zhang et al., 2020; De Bruijn et al., 2011; Li and Huang, 2016; Yehia et al., 2019; Mulyar and McInnes, 2020) in the text-mining field make an effort to alleviate the clinician’s burden according to exploit how to better utilize the knowledge contained in scientific discourse, accessible in the form of natural human language. Automating the process of understanding the relevant parts of the scientific literature allows for effective searching, and enabling inference of new information and hypothesis generation for clinical research.

\* Corresponding author

Counts and measurements are an important part of information source from the scientific discourse (Harper et al., 2021). However, extracting these count and measurement entities and their interactions is challenging, since the way scientists write them can be ambiguous and inconsistent, and the location of this information relative to the measurement can vary greatly.

In this paper, we focus on the SemEval 2021 MeasEval task which is composed of five sub-tasks that cover span extraction, classification, and relation extraction. As shown in Figure 1 it firstly demands to identify all the quantity spans given a paragraph from a scientific text. For each identified quantity, we need to extract the measured entity, measured property and qualifier which are corresponded to identified quantity. Besides, the relationships between quantity, measured entity, measured property and qualifier are also required to be identified. Lastly, the unit and type of the quantity is also needed to be recognized if they exist.

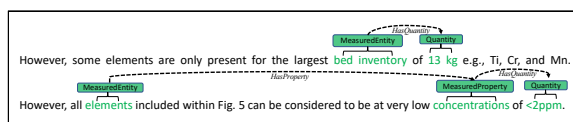


Figure 1: A sample of annotated snippet of dataset.

To tackle this challenging task aforementioned, we propose CONNER, a **cascade count and measurement** extraction tool, of which it primarily contains four components: (1) Model encoder gains the representation of both scientific paragraph text and the entities. (2) Quantity tagger extracts all the potential quantity entities within the paragraph. (3) Relation-specific object tagger recognizes the possible measured entities, measured properties and qualifiers, as well as their mutual relations. (4) Unit and modifier extractor identifies units and modifier by both rule-based approach and a simple classifier.

The rest of the paper is organized as follows. In Section 2, we elaborate on the whole workflow of CONNER and a detailed analysis of our experiments and results in Section 3. The paper is summarised and concluded in Section 4.

## 2 Model Description

### 2.1 Overview

The goal of CONNER is designed to identify all possible aspects of quantity items, including `Quantity`, `MeasuredEntity`, `MeasuredProperty`, `Qualifier`, `Unit` and `Modifier`, as well as their relations.

Inspired by (Wei et al., 2019), we assign a cascade framework that models relations as functions that map quantity to other objects in a sentence, which is contrary to the conventional prospective of relational triple extraction task (Gupta et al., 2016; Adel and Schütze, 2017; Zeng et al., 2018; Zheng et al., 2017) that firstly identity all the possible entities, then predict their relations. Based on our observation, there are roughly 90% entities overlapped in our tasks, which can be tackled smoothly by a cascade framework. Besides, our proposed method can just generate a limited amount of negative samples since there are only three types of relations in our tasks, which further prevent from the long-tail problem (Zhang et al., 2019; Li et al., 2020).

The basic idea of CONNER is a two-step pipeline model as shown in Figure 2. We firstly deploy a quantity tagger to identity all the possible `Quantity` from input paragraph in Section 2.2, for each predicted `Quantity`, we check all the potential relations to see if a relation can associate `MeasuredEntity`, `MeasuredProperty` and `Qualifier` with the `Quantity` in Section 2.4. In contrast of utilizing the proposed cascade framework, we adopt a rule-based method and a simple classifier in terms of extraction of unit and modifier in Section 2.5. We describe the detailed workflow below.

### 2.2 Model Encoder

The model encoder aims at gaining the semantic representations  $H$  of input paragraph text  $X$ , which will be further used in the following tagging module. In terms of the input paragraph that exceeds our pre-defined maximum length, we split them into pieces via full stop and encoder them separately. We experiment both ROBERTA (Liu

et al., 2019) and BERT model (Devlin et al., 2018) to encode the context information. We adopt  $H = \text{Encoder}(X)$  for brevity, and  $L$  denotes the length of the input paragraph.

### 2.3 Quantity Tagger

The lower level tagging model shown in Figure 2 is designed to predict the entire potential quantities in the input paragraph, which is an ensemble model incorporating a CRF layer and a PointerNet Layer. We illustrate the whole workflow as follows.

**PointerNet layer.** Driving from the PointerNetwork (Vinyals et al., 2015), two identical binary classifiers are adopted to detect the start and end position of quantities respectively. Each token is fed into the binary classifiers to predict whether the current token is aligned to a start or end position of a quantity span. Formally, given a contextual representation  $h_i \in H$ , we have:

$$p_i^{start} = \sigma(\mathbf{W}_{start}h_i + \mathbf{b}_{start}) \quad (1)$$

$$p_i^{end} = \sigma(\mathbf{W}_{end}h_i + \mathbf{b}_{end}) \quad (2)$$

where  $\mathbf{b}$  is the bias matrix and  $\sigma$  is the sigmoid activation function.  $p_i^{start}$  and  $p_i^{end}$  denotes the probability of identifying the  $i$ -th token in the input paragraph as the start or end position of a quantity, respectively. We set up a threshold score as 0.7, of which the current token will be assigned to 1 if its probability surpasses the threshold score, otherwise assigned to 0. The loss function of the quantity tagger is the following:

$$\mathcal{L}_{qt} = \frac{1}{L^2} \sum_{i=1}^L y_i^{start,end} \log P_i^{start,end} \quad (3)$$

where  $L$  denotes the length of the input paragraph,  $y_i^{start,end}$  is the ground truth label. In terms of multiple quantities appeared in the same paragraph, We adopt the same strategy as (Wei et al., 2019) that we adopt the nearest start-end pair match principle to decide the span of any quantity based on the results.

**CRF layer.** In this layer, we consider quantity recognition as a sequence-labeling problem. We select BIOS(Beginning, Inside, Outside, Single) as our label schema. Accordingly, given the representation sequence  $H = (h_1, h_2, \dots, h_L)$  we adapt a probability-based sequence detection conditional random field (CRF) model (Zheng et al., 2015),

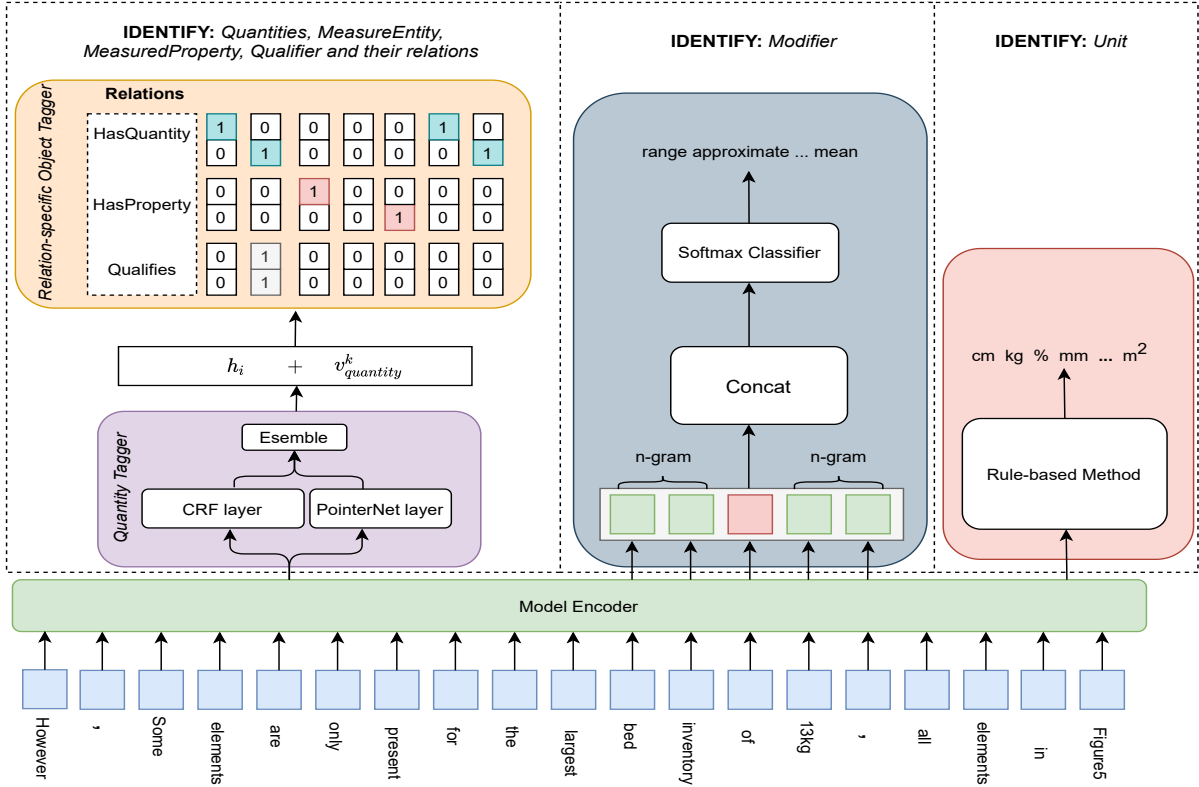


Figure 2: Overall model structure: the left dashed box corresponds to the Subsection 2.3 and 2.4, the right and middle ones correspond to Subsection 2.5.

which defines the conditional probability distribution  $P(Y|H)$  of label sequence  $Y$  given contextual word representation  $H$  aforementioned. We maximize the log-probability during training. In decoding, we set transition costs as infinite if it is invalid. The expected label sequence  $Y = (y_1, y_2, \dots, y_L)$  is predicted based on maximum scores in the decoding.

**Ensemble.** The experimental results are relatively comparable in terms of CRF layer and pointerNet layer (See Table 2.3). However, an empirical observation on the predicted results suggest that CRF layer tends to extract shorter spans, while PointerNet layer does the opposite. Presumably, the ensemble model can gain better results since the distribution of entities in the dataset exists both long and short spans.

We deploy our model ensemble considering the predicted quantities that are partly overlapped. To be more specific, we firstly obtain the predicted quantities from PointerNet as our final result, besides, if a predicted quantity from CRF layer strictly does not exist in the PointerNet result, i.e., there is no overlap between the two quantities, we add it to our final result as well.

## 2.4 Relation-specific Object Tagger

The upper level tagging module identifies the `MeasuredEntity`, `MeasuredProperty` and `Qualifier` as well as the involved relations with respect to the quantities obtained in the previous section. It consists of a set of relationship-specific taggers with the same structure as the quantity tagger in the lower-level for all possible relations. All object taggers identify the corresponding object for each detected quantity simultaneously.

Distinguish from the quantity tagger using representation vector  $h_i$  as input, the relation-specific object tagger also takes the quantity semantic features into account. Given each contextual representation of the current token  $h_i$ . The detailed tagging operations are as follows:

$$\tilde{p}_i^{start} = \sigma(\mathbf{W}_{start}^r(h_i + \mathbf{v}_{quantity}^k) + \mathbf{b}_{start}^r) \quad (4)$$

$$\tilde{p}_i^{end} = \sigma(\mathbf{W}_{end}^r(h_i + \mathbf{v}_{quantity}^k) + \mathbf{b}_{end}^r) \quad (5)$$

where  $\tilde{p}_i^{start}$  and  $\tilde{p}_i^{end}$  denotes the probability of predicting the start and end position of current token.  $\mathbf{v}_{quantity}^k$  represents the representation of  $k$ -th identified quantity via model encoder in Section 2.3.

We iterate all the possible relations across the given quantities. Accordingly, given the token representation  $h_i$  and quantity  $k$ , the loss function of relation-specific object tagger is as follows:

$$\mathcal{L}_{rot} = \frac{1}{L^2 \cdot R} \sum_{r=1}^R \sum_{i=1}^L y_{i,r}^{start,end} \log P_{i,r}^{start,end} \quad (6)$$

where  $r \in R$  denotes the  $r$ -th relation. The rest of symbol keeps the same as Equation 3. Note that tags  $y_i^{start,end} = 0$  if the objects are empty.

## 2.5 Unit and modifier extractor

**Unit extraction.** We design a set of rules to identify the units which are corresponding to each predicted quantity. The detailed description of schema is presented in Algorithm 1.

---

### Algorithm 1 Unit Method

---

**Require:** Quantity  $Q$ , which is a string of  $n$  characters.

- 1: The collection of all units that have appeared in train & trial:  $V$
  - 2:  $p = n$
  - 3: **for**  $i = 1$  to  $n$  **do**
  - 4:   **if**  $Q[i : n] \in V$  **then**
  - 5:     **return**  $Q[0 : i]$
  - 6:   **end if**
  - 7:   **if**  $Q[i]$  is a space **then**
  - 8:      $p = i + 1$
  - 9:   **end if**
  - 10: **end for**
  - 11:  $s = p$
  - 12: **while**  $s > 0$  and  $Q[s - 1]$  is a character **do**
  - 13:    $s = s - 1$
  - 14: **end while**
  - 15: **if**  $s > 0$  **then**
  - 16:    $p = s$
  - 17: **end if**
  - 18: **return**  $Q[p : n]$
- 

**Modifier Classification.** As none of the relations attached to the modifier in the input paragraph, we can not apply the relation-specific object tagger in terms of modifier extraction. Given a candidate quantity token  $x_i$ , we select its n-gram contextual tokens  $\{x_{i-n}, x_{i-n+1}, \dots, x_i, x_{i+1}, x_{i+n}\}$  and concatenate them as model input and then simply introduce a plain classifier to predict its labels:

$$c_i = \text{BERT}([x_{i-n}; \dots; x_i; \dots; x_{i+n}]) \quad (7)$$

$$y_i = \arg \max_{\theta} (\text{softmax}(c_i)) \quad (8)$$

where  $c_i$  denotes the representation of quantity and contextual tokens after BERT encoder and  $y_i$  is the predicted label of modifier in terms of current

token  $x_i$ . The training loss is the conventional cross entropy loss, we will not elaborate on it due to the space limit.

## 3 Experimental Results

### 3.1 Dataset

This SemEval evaluation has released the dataset online<sup>1</sup>, which includes a text file for each paragraph of scientific text along with annotations. As shown in Table 1, the overlapping entities are 9.3%, 0% and 90.7% in total of NEO, EPO and SEP in terms of train/dev/test set, respectively, which indicates the merit of applying CONNER to our tasks since it can naturally handle the overlapping entities.

	Train+Trial	Test
Sentence number	647	593
Avg. Sentence length	45	39
Max. Sentence length	200	304
Triples	2199	-
Cross Sentence Triples	65	-
NEO	203	-
EPO	0	-
SEO	1996	-

Table 1: Statistics of dataset, NEO represents none entity overlap, EPO represents entity pair overlap, SEP represents single entity overlap.

### 3.2 Experimental Settings

We adopt mini-batch mechanism to train our model with batch size as 8; the pretrained language model finetuning learning rate is set to  $2e-5$ , crf decoder learning rate is set to  $5e-3$ ; the hyper-parameters are determined on the validation set. We also complete words with wrong boundaries by design rules, e.g., "emain mostly neutral" in the raw text is corrected to "remain mostly neutral". The maximum length of sentence is set as 350. The number of n-gram is 0. We adopt Adam (Kingma and Ba, 2014) for optimization.

### 3.3 Main Experiments

The experimental results are conducted in test set, of which each entity category and relations are listed in Table 2. The result of extracting quantity outperforms the rest of entity categories by a large margin regarding named entity recognition. While the HasQuantity naturally achieves the best result in relation extraction task.

---

<sup>1</sup>The dataset is available at <https://github.com/harperco/MeasEval>

	Prec.(%)	Rec.(%)	F1(%)	F1 Over(%)
<b>Entities</b>				
Quantity	96.16	92.24	75.70	94.16
M.Entity	70.82	52.87	60.54	39.84
M.Prop.	72.79	56.71	63.75	43.66
Qualifier	20.00	12.32	15.25	0.0
Modifier	74.76	63.56	68.70	-
Unit	82.52	84.78	83.64	-
<b>Relations</b>				
HasQuan.	62.85	56.52	59.52	-
HasProp.	57.37	31.72	40.85	-
Qualifies	0	0	0	-
Overall	76.09	57.53	65.52	47.30

Table 2: Experimental results on test set, F1 Over represents F1 overlap. All results are produced by the official evaluation scripts.

### 3.4 Auxiliary Experiments

During the process of building our proposed system, we tested different schemes for each module of the our model and did relative experiments to compare their experiment results, the scheme with best performance is selected as our final modules consisting of CONNER. We present the in-depth analysis and experimental results listed below.

**Model encoder.** In Subsection 2.2, we separately adopt BERT-based and ROBERTA-based our model encoder. To examine the performance regarding different model encoder, we conduct experiments in the quantity identification stage for both identification of entities and relations. As we can notice in Table 3, ROBERTA-base all outperforms BERT-base so that it is selected as our final model encoder.

	Prec.(%)	Rec.(%)	F1(%)
<b>Entities</b>			
BERT-large	58.85	56.02	57.40
ROBERTA-large	60.37	57.68	58.99
<b>Relations</b>			
BERT-large	49.52	45.67	47.39
ROBERTA-large	49.52	52.94	51.17

Table 3: Experimental results of different model encoders

**Settings of ensemble scheme.** We tested the result of utilizing CRF layer and PointerNet layer independently, it shows comparable results as listed in Table 4. As we mentioned in in Subsection 2.3, combining the results of CRF and PointerNet can make the best use of both models, and results verified our assumption that ensemble models all outperform the singular models.

we also carried out two different ensemble approach for quantity tagger. The first one is as il-

lustrate in Subsection 2.3. The second approach is simpler: we take the union of the predicted quantities of CRF layer and PointerNet layer, and remove duplicate as our final prediction result. The experimental results in Table 4 suggested the first ensemble model achieve the best result, so that it is selected as our final ensemble scheme.

	Prec.(%)	Rec.(%)	F1(%)
CRF layer	60.37	57.68	58.99
PointerNet layer	59.67	56.53	58.06
Union ensemble	58.54	59.78	59.15
ensemble in Section 2.3	60.47	59.02	59.73

Table 4: Experimental results of different model encoders

**Settings of  $n$ -gram.** Different number of  $n$ -gram can affect the model performance to some extent, we thus tested introducing different length of context regarding extraction of the modifier. As shown in Table 5, the model achieves best performance with 45.13% F1 score when  $n$  is 0, meanwhile, we speculate the underlying reason is that model is not capable of capturing valid semantics from contextual tokens due to the limited amount of the modifiers in the whole dataset.

$n$ -gram	F1(%)
none context	<b>45.13</b>
window_char_5	42.22
window_char_10	41.84
window_word_1	44.63
window_word_3	44.08

Table 5: Experimental results of different model encoders

## 4 Conclusion

We proposed CONNER, a cascade count and measurement extraction tool to jointly identify the quantities and their attached items, as well as the corresponding relations for SemEval 2021 Task 8: Measurement. Our model extracts these entities and relations in a two-step pipeline method. We also exploited various of technical schemes during the competition and select the one that gains the best performance in the experiments, which help us win second-place in the final ranking.

## References

Heike Adel and Hinrich Schütze. 2017. [Global normalization of convolutional neural networks for joint entity and relation classification](#). In *Proceedings of the 2017 Conference on Empirical Methods in*

- Natural Language Processing*, pages 1723–1729, Copenhagen, Denmark. Association for Computational Linguistics.
- Taxiarchis Botsis, Michael D Nguyen, Emily Jane Woo, Marianthi Markatou, and Robert Ball. 2011. Text mining for the vaccine adverse event reporting system: medical text classification using informative feature selection. *Journal of the American Medical Informatics Association*, 18(5):631–638.
- Jiarun Cao, Chongwen Wang, and Liming Gao. 2018. A joint model for text and image semantic feature extraction. In *Proceedings of the 2018 International Conference on Algorithms, Computing and Artificial Intelligence*, pages 1–8.
- Berry De Bruijn, Colin Cherry, Svetlana Kiritchenko, Joel Martin, and Xiaodan Zhu. 2011. Machine-learned solutions for three stages of clinical information extraction: the state of the art at i2b2 2010. *Journal of the American Medical Informatics Association*, 18(5):557–562.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Pankaj Gupta, Hinrich Schütze, and Bernt Andrassy. 2016. [Table filling multi-task recurrent neural network for joint entity and relation extraction](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2537–2547, Osaka, Japan. The COLING 2016 Organizing Committee.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Peng Li and Heng Huang. 2016. Clinical information extraction via convolutional neural network. *arXiv preprint arXiv:1603.09381*.
- Yang Li, Tao Shen, Guodong Long, Jing Jiang, Tianyi Zhou, and Chengqi Zhang. 2020. Improving long-tail relation extraction with collaborating relation-augmented attention. *arXiv preprint arXiv:2010.03773*.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Andriy Mulyar and Bridget T McInnes. 2020. Mt-clinical bert: Scaling clinical information extraction with multitask learning. *arXiv preprint arXiv:2004.10220*.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *arXiv preprint arXiv:1506.03134*.
- Zhepei Wei, Jianlin Su, Yue Wang, Yuan Tian, and Yi Chang. 2019. A novel cascade binary tagging framework for relational triple extraction. *arXiv preprint arXiv:1909.03227*.
- Engy Yehia, Hussein Boshnak, Sayed AbdelGaber, Amany Abdo, and Doaa S Elzanfaly. 2019. Ontology-based clinical information extraction from physician’s free-text notes. *Journal of biomedical informatics*, 98:103276.
- Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. [Extracting relational facts by an end-to-end neural model with copy mechanism](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia. Association for Computational Linguistics.
- Ningyu Zhang, Shumin Deng, Zhanlin Sun, Guanying Wang, Xi Chen, Wei Zhang, and Huajun Chen. 2019. Long-tail relation extraction via knowledge graph embeddings and graph convolution networks. *arXiv preprint arXiv:1903.01306*.
- Yuanzhe Zhang, Zhongtao Jiang, Tao Zhang, Shiwan Liu, Jiarun Cao, Kang Liu, Shengping Liu, and Jun Zhao. 2020. Mie: A medical information extractor towards medical dialogues. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6460–6469.
- Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. 2015. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537.
- Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. [Joint extraction of entities and relations based on a novel tagging scheme](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1227–1236, Vancouver, Canada. Association for Computational Linguistics.
- Xuezhong Zhou, Yonghong Peng, and Baoyan Liu. 2010. Text mining for traditional chinese medical knowledge discovery: a survey. *Journal of biomedical informatics*, 43(4):650–660.



# Stanford MLab at SemEval-2021 Task 8: 48 Hours Is All You Need

Patrick Liu, Niveditha Iyer, Erik Rozi, Ethan A. Chi

Stanford University

{pliu1, nivsiyer, erikrozi}@stanford.edu, ethanchi@cs.stanford.edu

## Abstract

This paper presents our system for the Quantity span identification, Unit of measurement identification and Value modifier classification subtasks of the MeasEval 2021 task. The purpose of the Quantity span identification task was to locate spans of text that contain a count or measurement, consisting of a value, usually followed by a unit and occasionally additional modifiers. The goal of the modifier classification task was to determine whether an associated text fragment served to indicate range, tolerance, mean value, etc. of a quantity. The developed systems used pre-trained BERT models which were fine-tuned for the task at hand. We present our system, investigate how architectural decisions affected model predictions, and conduct an error analysis. Overall, our system placed 12 / 19 in the shared task and in the 2nd place for the Unit subcategory.

## 1 Introduction

The growing ease of access to large bodies of scientific information and research has not been accompanied by an improvement in ease of analysis or understanding of scientific text. While the performance of natural language processing tasks such as part-of-speech (POS) tagging and dependency parsing have seen improvements, analyzing counts and measurements in scientific texts has remained a largely unaddressed problem, though some work has been done by (Berrahou et al., 2013).

Measurements involve quantification (in units such as litres or kilograms), entities and the measured properties (e.g. growth rate of a fungus) and value modifiers (e.g. mean, range, tolerance). Extracting semantic relations between quantities, entities, value modifiers, and units of measurement and deriving meaning from those components is challenging because scientific communication can be ambiguous or inconsistent. In addition, the loca-

tion of this information relative to the measurement tends to vary.

The precise extraction and contextualization of measurements could enable accurate summarization of large bodies of scientific literature. Additionally, this could allow for unordered measurement and numeric data from scientific texts to be transformed into standardized ordered data for easier analysis.

In this paper, we describe a rapidly implemented, lightweight model that extracts measurement context in natural language. Our pipeline was implemented in approximately 48 hours by the first three authors, who had minimal formal neural NLP experience. Despite this, we were able to achieve strong results in the two categories (*Quantity* and *Unit*) that we entered, achieving second place in the *Unit* category and 12 / 19 overall.

## 2 Background

### 2.1 MeasEval Task Setup

Data for this task is made available through the MeasEval repository (Harper et al., 2021). Input data are formatted as a paragraph of scientific text in English from which measurements would be extracted. Text annotations are provided in the BRAT Annotation format (Stenetorp et al., 2012) and evaluated in a tab separated value format.

These annotations are provided in four types of spans. A **Quantity** contains a measurement of an entity, such as *300 ml* or *10%*. **Quantities** are associated with their respective *Unit*, such as *ml* or *%* respectively. Each **Quantity** can also contain additional **Modifiers** which will be outlined later. A **MeasuredEntity** contains the entity the **Quantity** is referring to, such as *a flask* or other object. Subtasks 3-5 also include identifying **MeasuredEntities**, **MeasuredProperties** and **Qualifiers**, though due to time constraints, this paper mainly focuses

on identifying **Quantities** and **Units** (subtasks 1 and 2).

For example, given the sentence “*However, some elements are only present for the largest bed inventory of 13 kg e.g., Ti, Cr, and Mn.*”, an identifiable **Quantity** is *13 kg*, with the **Unit** being *kg*. Each paragraph can have multiple quantities, modifiers and units associated with them.

## 2.2 Prior Work

The system this paper outlines is based on the BERT architecture (Devlin et al., 2019). BERT, which stands for Bidirectional Encoder Representations from Transformers, employs the self-attention based Transformer mechanism described in (Vaswani et al., 2017). BERT allows for pre-trained bidirectional representations to be used for each word token, which can then be applied to a wide variety of tasks. As such, only a limited amount of fine-tuning is required to model NLP tasks including sentence classification and sequence tagging.

One NLP task that is especially relevant for the Unit of Measurement identification task is sequence tagging and Named Entity Recognition (NER). This task involves identifying a span of text (such as a person or location) from a given body of text. There has been significant research in the past dedicated towards NER (Li et al., 2018). (Devlin et al., 2019) also addresses sequence tagging directly in their original paper describing BERT. Therefore, we found that applying BERT towards the shared task would prove transferable and effective.

## 3 System Overview

### 3.1 Quantity Identification

Since subtask 1, quantity span identification, involves the classification of individual phrases, we fine-tuned a pre-trained BERT-Large model (Devlin et al., 2019) to perform token-level classification on an IO labeling strategy based on the IOB labeling scheme (Ramshaw and Marcus, 1995). Each token in the training and trial sets is labeled as “I” if it is inside a labeled quantity, and “O” if it is outside the labeled quantity. Since consecutive entities are very rare, we do not use the “B” label. Such an encoding scheme is common for named-entity recognition. Tokenization is implemented using BertTokenizerFast from the Hugging Face

Transformers library.<sup>1</sup>

As with the tokenizer, we used Hugging Face library for a pre-trained BERT model. Instead of using the base BERT hidden layers (BERTModel) and manually building a few fine-tuning layers for classification, we decided to use BERT’s built-in BERTForTokenClassification class, which adds a single linear layer for classification on top of the hidden layers. This choice was made in the interest of saving programming time; we did not believe that adding multiple fine-tuning layers would cause a significant increase in accuracy to justify the time that would be spent. The model outputs labels for each of the tokens, labeling them with the aforementioned IO labeling scheme. Sequences of consecutive “I” labels are then recombined into segments of text, which we use as our predicted quantities.

We trained this model on a cross entropy loss metric, with loss on “I” labels upweighted by a factor of K to account for the low relative number of quantity tokens versus non-quantity tokens. We optimized on minibatches using the Adam optimizer (Kingma and Ba, 2015), and applied a cosine-annealing scheduler (Loshchilov and Hutter, 2016) to scale the down learning rate. Both of these have been shown to be effective in fine-tuning pretrained Transformer language models.

### 3.2 Unit Identification

Once a quantity had been identified, the next step involved identifying the respective unit for each quantity. For consistency, we once again used a BERT tokenizer to achieve this task. Intrinsically, unit identification is considered a NER task; therefore, by using Hugging Face Transformers and the BERTForTokenClassification class, we implemented a quick, yet robust method for identifying units.

The Unit Identification approach is essentially the same as the quantity identification approach: the model labels tokens as either “I” or “O” based on whether each token is considered to be inside or outside a quantity. This approach proved to be ubiquitous and simple to implement. We identified that tokens for units were consecutive, so this model was tasked with searching for a specific word or words that could be identified as units. For example, given the Quantity *20%*, the *%* token would

<sup>1</sup>[https://huggingface.co/transformers/model\\_doc/bert.html](https://huggingface.co/transformers/model_doc/bert.html)

be labeled as an “I” while all other tokens would be labeled as “O”. By implementing a consistent approach, we tested the capabilities of pretrained robust models such as BERT, which proves to be useful for creating robust models with low computational resources.

### 3.3 Modifier Multilabel Classification

In the final part of subtask 2, we were tasked to classify Quantity spans by “Modifiers.” Unlike the previous two parts of the pipeline, we approached this task as a multilabel classification task. Each quantity could be associated with multiple labels, such as *IsRange* and *IsApproximate*, or no labels at all. This model mapped out each label as an individual class, such that there were 10 total classes.

To predict the likelihood that a Quantity span is associated with a class, we first tokenized each quantity using BERT. These tokens were then passed into a simple one layer dropout and linear model which output 10 features. From here, each feature was considered as an independent binary classification problem in order to predict multiple classes. For simplicity, any logit greater than 0.5 was predicted as a “true” value. Binary Cross Entropy loss was used as the criterion for training.

This is a simple, yet effective approach. As this approach uses the same BERT models as the first two sections, with the key difference being the classification problem itself, we demonstrate the versatility of BERT for these tasks. Once again, we used the Hugging Face Transformers library for a pre-trained BERT model, which allowed for quick deployment. The results obtained in later sections highlight how quick and inexpensive models can still obtain acceptable results.

## 4 Experimental Setup

For all of the subtasks, we used the provided training split of 2531 annotations across 249 texts versus 832 annotations across 66 texts.

Besides tokenizing the text and applying IO labels (see section 3.1) to get the data into a format fit for the BERT token classifier, we did not preprocess the data significantly. We tested filtering unrecognizable and irrelevant characters using regex but found no meaningful improvement in performance. We thus decided to omit preprocessing to keep with the simplistic theme of this paper.

During the hyperparameter tuning phase, we compared models based on token-level F1 score,

implemented using the classification report from the metrics module of PyTorch, yielding results consistent with the provided testing script, which evaluated on the F1 overlap score. Due to the limit on the total number of iterations we could go through due to the deadline, we tuned hyperparameters by hand. We found that out of the hyperparameters we tuned, the scheduler hyperparameters were irrelevant because the scheduler was rarely used at all. The number of training epochs likewise did not come into play so long as we let the training continue until trial loss stopped decreasing; around 15 epochs was almost always enough. The only hyperparameters that seemed to significantly impact training results were the learning rate and the weight difference between quantity and non-quantity labels. After some manual tuning, we found that a learning rate of around  $10^{-5}$  and no weight-difference adjustment performed best on the trial set for this task, resulting in our final token-level quantity identification F1 score of 0.85 on the trial set.

For the quantity identification subtask, several post-processing steps were implemented to correct simple errors in our model’s predictions, based on common errors detected via manual error analysis in the model’s predictions. First, commas and spaces were removed from the ends of the predicted quantities to correct for potential small character offsets on the edges of predictions, from either model error or the tokenization-detokenization process. Furthermore, all quantities that did not contain either numeric characters or strings that commonly occur in numbers (such as *two* or *-teen*) were discarded from the prediction pool. Additionally, all prepositions from a hand-picked set like *beyond* and *at* were cut from the front of each input string in order to better isolate numeric quantities from relational terms. The trial set results with and without post-processing are shown in Table 1. All the post-processing steps reduce the amount and length of predicted quantity spans. Therefore, the significantly higher precision score after post-processing indicates that the base model tends to overpredict identified quantities.

## 5 Results

Our model’s overall precision, recall, and F1 score on overlap on the trial set are as shown in Table 1. The model’s performance on evaluation metrics across the subtasks are shown in Table 2.

Metric	With	Without
Precision	0.8153	0.564
Recall	0.2912	0.301
F1 (overlap) score	0.2615	0.211

Table 1: Trial set results with and without post-processing

Category	F1 score	Ranking
Overall	0.272	12
Quantity	0.818	10
Unit	0.76	2
Modifier	0.408	8

Table 2: Evaluation set results and ranking (out of 19)

These results are promising given the speed of implementation of our system. Across all 5 sub-tasks, our final F1 score was 0.272. Overall, the model ranked 12th out of 19 teams in the evaluation phase of MeasEval: Counts and Measurements.

## 6 Conclusion

We propose a BERT-based model for information extraction of measurements and their contextual qualifiers from text. As scientific texts become increasingly open for public consumption, we hope that such systems will help present findings to broader audiences in an accessible manner. Going forward, possible areas of exploration include replacing BERT with more powerful models and augmenting training data that the model frequently makes erroneous predictions on.

## Acknowledgments

This research effort would not have been possible without the support of Stanford ACM Lab. The authors thank Corey Harper, Jessica Cox, Ron Daniel, Paul Groth, Curt Kohler and Antony Scerri for organising SemEval 2021 Task 8: MeasEval - Counts and Measurements. We also thank Jillian Tang for helpful discussions.

## References

Soumia Lilia Berrahou, Patrice Buche, Juliette Dibié-Barthélemy, and Mathieu Roche. 2013. [How to extract unit of measure in scientific documents?](#) In *KDIR/KMIS 2013 - Proceedings of the International Conference on Knowledge Discovery and Information Retrieval and the International Conference on Knowledge Management and Information Sharing*,

*Vilamoura, Algarve, Portugal, 19 - 22 September, 2013*, pages 249–256. SciTePress.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. [Adam: A method for stochastic optimization](#). In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2018. [A survey on deep learning for named entity recognition](#). *CoRR*, abs/1812.09449.

Ilya Loshchilov and Frank Hutter. 2016. [SGDR: stochastic gradient descent with restarts](#). *CoRR*, abs/1608.03983.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. [Text chunking using transformation-based learning](#). *CoRR*, cmp-lg/9505040.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun’ichi Tsujii. 2012. [brat: a web-based tool for NLP-assisted text annotation](#). In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#). *CoRR*, abs/1706.03762.

# LIORI at SemEval-2021 Task 8: Ask Transformer for measurements

**Adis Davletov**

RANEPa, Moscow, Russia  
Lomonosov Moscow State University, Moscow, Russia  
davletov-aa@ranepa.ru

**Denis Gordeev**

RANEPa, Moscow, Russia  
gordeev-di@ranepa.ru

**Nikolay Arefyev**

Lomonosov Moscow State University, Moscow, Russia  
Samsung Research Center Russia, Moscow, Russia  
HSE University, Moscow, Russia  
nick.arefyev@gmail.com

**Emil Davletov**

Katanov Khakas State University, Abakan, Russia  
edavletov@yandex.ru

## Abstract

This work describes our approach for subtasks of SemEval-2021 Task 8: MeasEval: Counts and Measurements which took the official first place in the competition. To solve all subtasks we use multi-task learning in a question-answering-like manner. We also use learnable scalar weights to weight subtasks' contribution to the final loss in multi-task training. We fine-tune LUKE to extract quantity spans and we fine-tune RoBERTa to extract everything related to found quantities, including quantities themselves.

## 1 Introduction

SemEval-2021 Task 8 consisted of five subtasks that covered span extraction, classification, and relation extraction tasks. This paper presents solutions to all five of them which showed the best results in the competition<sup>1</sup>.

In the subtask 1(A) participants were asked to retrieve *Quantity* (**Q**) spans from texts. For example, in the following text **"The soda can's volume was 355 ml."**, the system should retrieve **"355 ml"** as **Q** span. The rest of the subtasks were to extract information related to retrieved *Quantities* (**Qs**) from subtask A.

The subtask 2(B) was to extract the *Unit of measurement* (**UoM**) of the extracted **Q** and also to classify it into 10 classes: *HasTolerance*, *IsApproximate*, *IsCount*, *IsList*, *IsMean*, *IsMeanHasTolerance*, *IsMeanIsRange*, *IsMedian*, *IsRange*, *IsRangeHasTolerance*. It should be noted that some

<sup>1</sup><https://github.com/davletov-aa/meas-eval>

**Qs** could be related to more than one type and there were ones which didn't belong to any type. The subtask 3(C) was to extract *Measured Entity* (**ME**) and *Measured Property* (**MP**) spans. In the subtask 4(D) additional *Qualifier* (**Qlfr**) spans, which helped to validate or understand the extracted **Q**, were asked to be extracted. And finally, subtask 5(E) was to extract relations between **Qs**, **MEs**, **MPs** and **Qlfrs**.

More detailed information about the competition could be found in the Harper et al. (2021)'s shared task description paper.

## 2 Related Work

Span extraction and classification problems have a long history of studies and are often studied as a part of Named Entity Recognition (NER). For example, the NER dataset Ontonotes v5 (Weischedel et al., 2013) contains such entities as "Quantity", which also includes measurements, and "Money". However, the general NER approach used in Ontonotes or ConLL 2003 (Sang and De Meulder, 2003) datasets is not so fine-grained as the one that is used in the task under study.

Most state-of-the-art models for named entity recognition and relation extraction are based on Transformer architecture by Vaswani et al. (2017). For example, the top three best models for Ontonotes v5 according to paperswithcode.com use BERT<sup>2</sup>. BERT is a large pre-trained language model based on Attention (Devlin et al., 2019).

<sup>2</sup><https://paperswithcode.com/task/named-entity-recognition-ner>

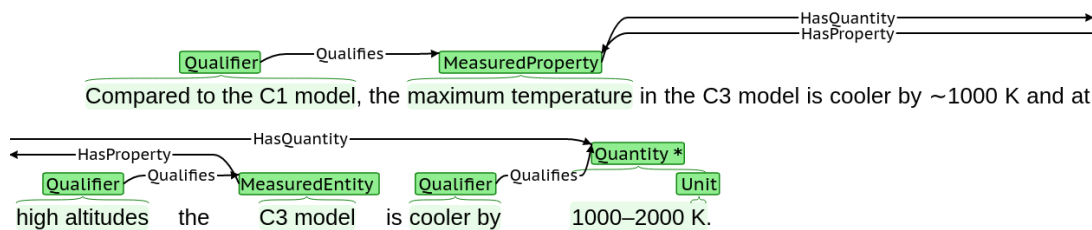


Figure 1: Data example. It shows that one named entity may have several incoming and outgoing relations.

BERT has a unique training procedure where the model is trained using Masked language objective, where some tokens are replaced with a special '[MASK]' token and the model should predict the original token. BERT also had an additional training objective - the model had to predict whether a sentence was random or it followed the first sentence. However, some papers have investigated that BERT is undertrained and that training BERT on more data and for a longer time might increase model performance. RoBERTa was one of the first and more influential papers of such kind (Liu et al., 2019). RoBERTa modifies BERT’s pretraining procedure. The RoBERTa model is trained longer, with bigger batches over more data and on longer sequences. RoBERTa’s authors have also found that removing the next sentence prediction objective from BERT matches or slightly improves BERT performance. Researchers have also suggested ways of leveraging the nature of the task and adding some problem bias to named entity recognition. Among such works, which is currently the best performing for Ontonotes v5 and CoNLL 2003 according to paperswithcode.com, is LUKE (Yamada et al., 2020). The authors of LUKE have added a new language modeling task that consists of predicting randomly masked words and entities in an entity-annotated corpus retrieved from Wikipedia. The authors have also expanded the self-attention mechanism to entity types and consider entity types when computing attention scores. The proposed approach allowed the authors to achieve state-of-the-art results not only for named entity recognition but for a bunch of other unrelated tasks such as SQuAD1.1 question answering.

For relation extraction, Transformer-based models also outperform other approaches. A promising approach is treating relation extraction as a question answering problem. Among works implementing this approach, we can mention (Cohen et al., 2020) where the authors restructured relation classification as a Question answering (QA) like span

prediction problem. It allowed them to get state-of-the-art results for TACRED and SemEval 2010 task 8 datasets.

### 3 System Description

#### 3.1 Data

The data provided by the organizers contained plain text files and their annotations in tsv format. There were in total 248 training texts, 65 trial ones, and 135 for the evaluation phase. There were 2764, 897, and 1620 annotated entities respectively. The files have been approximately equally distributed among several domains: Agriculture, Astronomy, Biology, Chemistry, Computer Science, Earth Science, Engineering, Materials Science, Mathematics, Medicine. Entities could have been labeled into 5 classes: **Q**, **ME**, **MP**, **UoM** or **Qlfr**.

As input data in the competition was in the form of plain text extracts, we first split them into sentences using PunktSentenceTokenizer and PunktTrainer from NLTK library (Bird et al., 2009). We trained PunktTrainer on texts from the training set. We did data augmentation by including text extracts consisting of two sentences following each other for each text document. So if we had original sentences  $[s_1, s_2, s_3, s_4]$  we get an augmented set of texts  $[s_1, s_2, s_3, s_4, s_1s_2, s_2s_3, s_3s_4]$ . Then we split each example into tokens using RegexpTokenizer from the NLTK library with the following  $\backslash w+ | \backslash ( | \backslash ) | \backslash [ | \backslash ] | [- | . | , | ] | \backslash S+$  regular expression. We used the train set for training and the trial set for development.

Also, we relabeled *Qualifier* to *QuantityQualifier* (**QQ**), *MeasuredEntityQualifier* (**MEQ**), and *MeasuredPropertyQualifier* (**MPQ**). By this little trick we solved the problem with examples having multiple **Qlfrs** corresponding to either **Q**, **ME**, or **MP**.

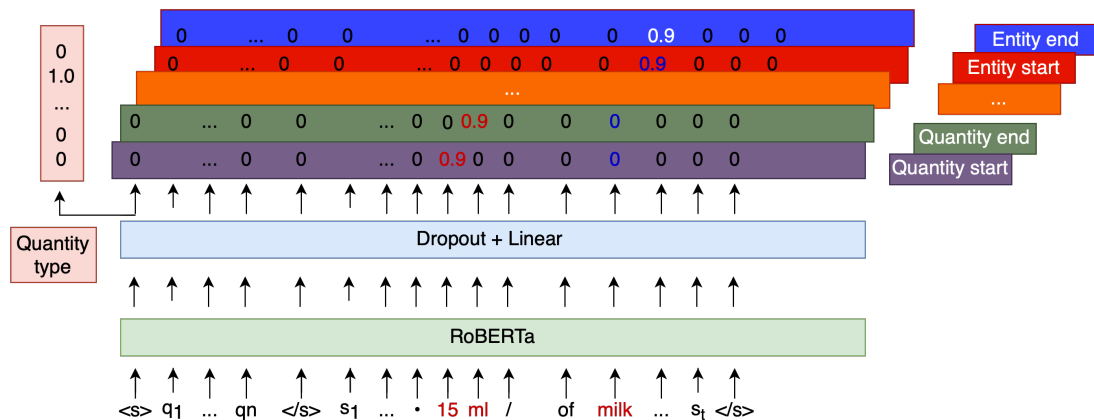


Figure 2: Architecture of the QuAnt system. It takes tokenized text with marked *MeasuredEntity* as input and predicts all needed spans and the class of the *Quantity*.

Hyperparam	Value
dropout	0.1
weight_decay	0.1
warmup_proportion	0.1
lr	1e-4
lr_scheduler	linear_warmup
optimizer	Adam
epochs	50
bs	128
max_seq_len	128
model	xlm-roberta-large
max_grad_norm	1.0
validate_per_epoch	4

Table 1: Training hyperparameters of **QuAnt** system, submitted to competition

### 3.2 QuAnt System

The architecture of the QuAnt<sup>3</sup> is shown in Figure 2. As could be seen, our model uses the RoBERTa model to extract features for each example. It solves all subtasks of the competition in a multi-task question answering way. We ask our model to predict all BPE subword-level start and end positions of spans (answers) related to **Q** (question). We ask the model by inserting special tokens “•” and “/” around **Q**. Also, the model makes multi-class multi-label predictions regarding the *type of the Quantity* (QT).

It takes text extracts containing some **Qs** and positions of the **Q** regarding which it should make predictions.

For example, for the input text “**The soda can’s**

<sup>3</sup>QuAnt - the system deals with quantities in a question-answering-like manner

volume was 355 ml.” and for subword-level positions (6, 7) of the **Q** “355 ml”, the model should predict the following start and end positions: (6, 7) for **Q** (A), (7, 7) for **UoM** (B), (3, 3) for **ME** (C), (4, 4) for **MP** (C), (2, 2) for **MEQ** (D). Also, the model shouldn’t predict any label for **QT** (B).

#### 3.2.1 Extract Quantities

So, our approach needs quantity span information as input. And to get that information we went with fine-tuning the LUKE model (Yamada et al., 2020) on the NER task to predict **Q** spans. We used the code provided by the authors of the model. We trained it on the augmented dataset in BIO format with the following hyperparameters: maximum-entity-length, maximum-sequence-length, learning rate, and batch size were set to 64, 256, 1e-5, and 4 respectively. We trained two models with the weight\_decay hyperparameter equal to 0.1 and 0.01 for 10 epochs. We were validating our models four times per epoch on the development set and saving the top 3 best checkpoints during training, resulting in a total of 6 models.

So after two training runs, we got 6 trained models. Using the development set we chose the best combination of them for a simple word-level voting ensemble.

#### 3.2.2 Extract Everything

During training and validation, we use **Q** spans from the annotated set. During test prediction, we use spans predicted by the ensemble of quantity extractors from the previous section. Because of our test time augmentation process, we had been able to get up to three entries per each **Q**: for the sentence containing it and for it with either its left

or right context sentences.

We split tokenized examples into byte-pair-encoding (BPE) subwords with RoBERTaTokenizer which resulted in the following RoBERTa inputs marked by symbols "•" and "/" "Qs":

[CLS] {Optional question prefix [EOS]}  $s_1 \dots \cdot w / \dots s_n$  [EOS].

To vectorize **QT**, we use the output from the last layer corresponding to [CLS] token. And to predict start and end probabilities for each subword of each span type we use outputs from the last layer. We feed them to linear layers to predict **QTs** and span starts and span ends.

During training we optimize the following weighted loss:

$$L_{total} = -\frac{w_{QT}}{bs} \sum_{i=1}^{bs} QT_k^i * \log(\hat{QT}_k^i) - \sum_{ST \in ST_s} \left( -\frac{w_{ST}}{2bs} \left( \sum_{i=1}^{bs} ST_{start}^i * \log(\hat{ST}_{start}^i) + \sum_{i=1}^{bs} ST_{end}^i * \log(\hat{ST}_{end}^i) \right) \right)$$

, where  $bs$  – batch size,  $[w_{QT}; w_{ST} | ST \in ST_s] = softmax([w_{qt}; w_{st} | st \in ST_s])$  – learnable weights vector initialized with ones,  $QT^i$  – one-hot encoded **QT** of  $i$ -th example (which could be zero vector in some cases and will not contribute during training),  $\hat{QT}^i$  – predicted **QT** probability distribution,  $ST_s$  – set of following span types: [**Q**, **ME**, **MP**, **Qlfr**, **UoM**, **QQ**, **MEQ**, **MPQ**],  $ST_{start}^i$  – one-hot encoded start position of the corresponding span.  $\hat{ST}_{start}^i$  – predicted start positions probability distribution. The same goes for  $ST_{end}^i$ .

We trained our model without adding an optional question prefix to RoBERTa inputs. We used hyperparameters from Table 1.

As our test predictions include duplicated predictions for the same **Q** due to the test time augmentation, we remove identical predictions. Worth noting, that there still might be duplicates left in the case of different extracted values for the same **Quantity**. Because of this, our submitted results are higher than the results without test time data augmentation.

So, our model takes **Q** with its context as input and predicts its type and extracts various spans. For all of the subtasks except the subtask E we treated extracted answers as is. For the subtask E we used

the following rules to extract relations between **Q**, **MP**, **ME** and **Qlfr** (**QQ**, **MEQ**, **MPQ**):

- (**MP**, *HasQuantity*, **Q**);
- if there is **MP** then (**ME**, *HasProperty*, **MP**), otherwise (**ME**, *HasQuantity*, **Q**);
- (**QQ**, *Qualifies*, **Q**), (**MEQ**, *Qualifies*, **ME**), (**MPQ**, *Qualifies*, **MP**);

## 4 Experiments and Results

In this section, we report the results of our post-evaluation experiments.

First, we experimented with base models. We tried different subtask weighting strategies. As we solve the task in a multi-task way, we need to aggregate the losses of each subtask to optimize the final loss. And here, we tried to just average them (**equal**) or take the weighted sum using learnable weights (**softmax**, **rsqr+log**) vector **W** with the length equal to the number of training subtasks. In the case of **softmax** weighting strategy we just use softmax over the vector **W**. In the case of **rsqr+log**, we divide each subtask’s loss to its squared learnable weight and sum with the logarithm of it. This approach of weighting subtasks in multi-task learning was introduced by Kendall et al. (2018).

We also experimented with data augmentation. But unlike experiments we did in the evaluation period, here we didn’t do test time data augmentation.

Also, we tried to concatenate the question prefix to an input example. We experimented with prefix *Find measured entities and properties of marked quantity*. We hoped it could give extra information to the model regarding the nature of the answer.

Table 2 shows the best results for the development dataset and Table 3 shows corresponding results for the test dataset. Also, there are our official submission results.

Table 2 shows that training time data augmentation improves the overall score. Also, we could see that including prefix question did not improve the overall scores of the models which use data augmentation. Yet we see the opposite picture for the test set in Table 3. It can also be seen that RoBERTa-large not necessarily outperforms RoBERTa base.

We see that using just the average sum of subtask’s losses demonstrates the best results.

We also tried to fine-tune the large version of XLM-R with the best hyperparameters from base



Model	Aug	WSch	PQ	O	Q	ME	MP	Qlfr	UoM	M	HQ	HP	Qlfs
post-evaluation phase results: roberta.base													
QA-v1	F	equal	F	45.2	98.9	32.5	36.2	15.2	73.9	66.1	42.4	21.2	9.3
QA-v2	F	equal	T	45.6	98.9	33.2	36.1	15.7	74.3	73.0	42.9	22.6	11.3
QA-v3	F	rsqr+log	F	45.3	98.9	32.3	35.4	15.2	74.7	70.1	41.8	22.0	<b>12.9</b>
QA-v4	F	rsqr+log	T	45.8	98.9	33.6	<b>37.8</b>	13.4	73.0	67.9	<b>46.1</b>	22.5	9.7
QA-v5	F	softmax	F	45.8	98.9	33.4	36.6	13.5	74.0	72.6	42.3	20.5	11.8
QA-v6	F	softmax	T	45.6	98.9	32.5	37.3	16.2	<b>75.3</b>	75.5	45.6	21.8	11.3
QA-v7	T	equal	F	<b>47.7</b>	98.9	33.1	35.6	16.1	74.3	77.7	40.3	22.7	9.6
QA-v8	T	equal	T	46.1	98.9	32.8	35.5	11.1	73.7	76.4	38.9	21.2	9.3
QA-v9	T	rsqr+log	F	47.6	98.9	32.4	36.8	<b>18.7</b>	73.9	<b>78.3</b>	40.0	21.9	12.6
QA-v10	T	rsqr+log	T	46.1	98.9	33.1	36.3	14.3	73.9	78.2	42.0	22.3	10.2
QA-v11	T	softmax	F	47.3	98.9	32.9	37.2	16.6	73.6	78.1	41.8	23.0	10.7
QA-v12	T	softmax	T	46.9	98.9	<b>34.5</b>	35.2	13.6	73.6	76.3	39.9	<b>24.2</b>	10.0
post-evaluation phase results: roberta.large													
QA-v1	T	equal	F	<b>49.3</b>	98.6	37.8	38.3	17.7	<b>75.6</b>	78.1	43.7	27.0	<b>10.3</b>
QA-v2	T	rsqr+log	F	48.8	98.5	35.0	<b>41.3</b>	10.7	75.3	77.7	<b>46.0</b>	24.5	6.9
QA-v3	T	softmax	F	48.9	98.5	<b>37.9</b>	38.1	<b>17.8</b>	73.7	<b>78.4</b>	44.4	<b>27.2</b>	<b>10.3</b>

Table 2: Best Overlap F1 scores for the dev set. Aug – augmentation, WSch – weighting Scheme, PQ – Prefix Question, O - Overall, Q – Quantity, ME – Measured Entity, MP – Measured Property, Qlfr – Qualifier, UoM – Unit, M – Modifier, HQ – Has Quantity, HP – Has Property, Qlfs – Qualifies.

Model	Aug	WSch	PQ	O	Q	ME	MP	Qlfr	UoM	M	HQ	HP	Qlfs
evaluation phase results: roberta.large													
QA-v1	T	softmax	F	51.9	86.1	43.7	46.7	16.3	72.2	64.2	48.2	31.8	9.2
evaluation phase results: best results of other competitors													
				47.3	85.5	40.6	43.7	10.7	80.4	61.4	42.4	25.7	6.4
post-evaluation phase results: roberta.base													
QA-v1	F	equal	F	44.8	84.7	38.8	38.1	15.4	66.9	52.0	39.5	24.3	8.5
QA-v2	F	equal	T	43.5	84.7	36.3	34.5	12.3	66.9	57.0	37.6	21.9	5.6
QA-v3	F	rsqr+log	F	45.1	84.7	39.3	39.7	11.9	67.2	50.9	41.7	24.3	7.8
QA-v4	F	rsqr+log	T	45.2	84.7	39.3	40.1	13.8	67.0	48.8	41.7	24.8	7.1
QA-v5	F	softmax	F	43.6	83.8	37.5	35.9	14.9	67.9	49.5	37.9	23.3	8.3
QA-v6	F	softmax	T	42.7	84.7	37.6	32.5	10.2	67.0	55.9	34.4	20.4	5.8
QA-v7	T	equal	F	44.6	84.2	37.4	37.9	9.8	67.4	57.2	39.7	23.6	6.2
QA-v8	T	equal	T	45.7	84.7	39.4	38.8	12.5	66.7	58.8	41.9	24.2	7.4
QA-v9	T	rsqr+log	F	45.9	84.4	39.0	38.3	<b>18.7</b>	66.1	58.7	41.5	24.1	<b>10.6</b>
QA-v10	T	rsqr+log	T	<b>47.1</b>	84.7	39.9	<b>42.0</b>	12.5	<b>68.2</b>	<b>59.6</b>	<b>44.4</b>	<b>26.6</b>	7.4
QA-v11	T	softmax	F	45.8	84.5	<b>40.0</b>	39.9	14.3	66.8	56.1	42.2	24.7	7.7
QA-v12	T	softmax	T	46.0	84.7	39.9	39.3	12.3	67.4	56.2	41.9	26.3	6.6
post-evaluation phase results: roberta.large													
QA-v1	T	equal	F	<b>48.9</b>	<b>84.6</b>	<b>43.0</b>	<b>45.6</b>	<b>15.4</b>	<b>67.0</b>	56.6	<b>47.7</b>	<b>32.0</b>	8.0
QA-v2	T	rsqr+log	F	47.2	83.8	41.9	42.0	9.2	66.7	58.0	44.6	29.7	6.0
QA-v3	T	softmax	F	47.6	<b>84.6</b>	41.9	41.8	<b>15.4</b>	66.5	<b>59.9</b>	44.9	29.7	<b>8.3</b>

Table 3: Overlap F1 scores for the test set. Aug – augmentation, WSch – weighting Scheme, PQ – Prefix Question, O - Overall, Q – Quantity, ME – Measured Entity, MP – Measured Property, Qlfr – Qualifier, UoM – Unit, M – Modifier, HQ – Has Quantity, HP – Has Property, Qlfs – Qualifies.

models. In the case of large models, again, **equal** weighting scheme demonstrated the best result.

In all our post-evaluation experiments we used the same settings as in Table 1. We tried learning rates from  $[5e - 5, 1e - 4, 2e - 4]$  and batch sizes from  $[32, 64, 128]$ .

## 5 Conclusion

In this paper, we introduced our solution to SemEval-2021 Task 8: MeasEval: Counts and Measurements. Our approach was based on RoBERTa and LUKE models. We show that extracting mea-

surements from a text can be treated as a question-answering task. In this work, we tried a set of different models, hyperparameters, and weighting schemes and present their effect on the final result.

## References

- Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. ” O’Reilly Media, Inc.”.
- Amir DN Cohen, Shachar Rosenman, and Yoav Gold-

- berg. 2020. Relation extraction as two-way span-prediction. *arXiv preprint arXiv:2010.04829*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of deep bidirectional transformers for language understanding**. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Corey Harper, Jessica Cox, Curt Kohler, Antony Scerri, Ron Daniel Jr., and Paul Groth. 2021. SemEval 2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In *Proceedings of the Fifteenth Workshop on Semantic Evaluation (SemEval-2021)*, Bangkok, Thailand (online). Association for Computational Linguistics.
- Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. **Attention is all you need**. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nanwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, et al. 2013. Ontonotes release 5.0 ldc2013t19. *Linguistic Data Consortium, Philadelphia, PA*, 23.
- Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention. In *EMNLP*.

# Sattiy at SemEval-2021 Task 9: An Ensemble Solution for Statement Verification and Evidence Finding with Tables

Xiaoyi Ruan, Meizhi Jin, Jian Ma, Haiqin Yang<sup>§</sup>, Lianxin Jiang,  
Yang Mo and Mengyuan Zhou

Ping An Life Insurance Co., Ltd.  
Shenzhen, Guangdong province, China

{RUANXIAOYI687, EX-JINMEIZHI001, MAJIAN446, JIANGLIANXIN769, MOYANG853}@pingan.com.cn

<sup>§</sup> the corresponding author, email: hqyang@ieee.org

## Abstract

Question answering from semi-structured tables can be seen as a semantic parsing task and is significant and practical for pushing the boundary of natural language understanding. Existing research mainly focuses on understanding contents from unstructured evidence, e.g., news, natural language sentences, and documents. The task of verification from structured evidence, such as tables, charts, and databases, is still less explored. This paper describes sattiy team’s system in SemEval-2021 task 9: Statement Verification and Evidence Finding with Tables (SEM-TAB-FACT). This competition aims to verify statements and to find evidence from tables for scientific articles and to promote the proper interpretation of the surrounding article. In this paper, we exploited ensemble models of pre-trained language models over tables, TaPas and TaBERT, for Task A and adjust the result based on some rules extracted for Task B. Finally, in the leaderboard, we attain the F1 scores of 0.8496 and 0.7732 in Task A for the 2-way and 3-way evaluation, respectively, and the F1 score of 0.4856 in Task B.

## 1 Introduction

Semantic parsing is one of the most important tasks in natural language processing. It not only needs to understand the meaning of natural language statements, but also needs to map them to meaningful executable queries, such as logical forms, SQL queries, and Python code (Pan et al., 2019; Lei et al., 2020; Zhu et al., 2021). Question answering from semi-structured tables is usually seen as a semantic parsing task (Pasupat and Liang, 2015), where questions are translated into logical forms that can be executed against the table to retrieve the correct denotation (Zhong et al., 2017).

Practically, it is significant in natural language understanding to verify whether a textual hypoth-

esis is entailed or refuted by evidence (Bentham, 2008; D. et al., 1978). The verification problem has been extensively studied in different natural language tasks, such as natural language inference (NLI) (Bowman et al., 2015), claim verification (Hanselowski et al., 2018), recognizing of textual entailment (RTE) (Dagan et al., 2005), and multi-model language reasoning (NLVR/NLVR2) (Suhr et al., 2018). However, existing research mainly focuses on verifying hypothesis from unstructured evidence, e.g., news, natural language sentences and documents. Research of verification under structured evidence, such as tables, charts, and databases, is still in the exploratory stage.

This year, SemEval-2021 Task 9: Statement Verification and Evidence Finding with Tables (SEM-TAB-FACT), aims to verify statements and find evidence from tables in scientific articles (Wang et al., 2021a). It is an important task targeting at promoting proper interpretation of the surrounding article.

The competition tries to explore table understanding from two tasks:

- Task A - Table Statement Support: The task aims to determine whether a statement is fully supported, refuted, or unknown to a given table.
- Task B - Relevant Cell Selection: given a statement and a table, the task is to determine which cells in the table provide evidence for supporting or refuting the statement.

The competition contains the following challenges:

- In task A, there is no training data for the “Unknown” category and the number of tables is small in the training set.
- The lexical expression of the table may be different from that in the statement.
- The table structure is complex and diverse. A table may contain missing values while the supporting evidence may be resided in cells across

several rows or columns.

- It is difficult to understand the statements. For example, some statements express totally different semantics meaning with only one different word. This difficulty makes it even harder to find the evidence cells from tables.

To overcome these challenges, we incorporate several key technologies in our implementation:

- developing a systematic way to generate data from the “Unknown” category;
- including additional data corpus to enrich the training data;
- exploiting existing state-of-the-art pre-trained language models over tables, TaBERT (Yin et al., 2020) and TaPas (Yin et al., 2020), and ensembling them into a powerful one;
- aligning contents in tables and statements while constructing manual rules for tackling Task B.

The test shows that our implementation can increase the performance according and finally, in the leadboard, we attain the F1 scores of 0.8496 and 0.7732 in Task A for the 2-way and 3-way evaluation, respectively, and the F1 score of 0.4856 in Task B.

The rest of this paper is organized as follows: In Sec. 2, we briefly depict related work to our implementation. In Sec. 3, we detail our proposed system. In Sec. 4, we present the experimental setup and analyze the results. Finally, we conclude our work in Sec. 5.

## 2 Related Work

Recently, pre-trained language models (PLMs), e.g., BERT (Devlin et al., 2019), XLNET (Yang et al., 2019), and RoBERTa (Liu et al., 2019), have witnessed the burgeoning of promoting various downstream NLP tasks, such as reading comprehension, named entity recognition and text classification (Li et al., 2020; Lei et al., 2021). However, the current pretrained language models are basically trained on the general text. They are not fit for some tasks, e.g., Text-to-SQL, Table-to-Text, which need to encode the structured data, because the data in the structured table also needs to be encoded at the same time. Directly applying the existing PLMs may face the problem of inconsistency between the encoded text from the table and the pretrained text.

TaBERT (Yin et al., 2020) is a newly proposed pretrained model built on top of BERT and jointly learns contextual representations for utterances and

the structured schema of database (DB) tables. This model views the verification task completely as an NLI problem by linearizing a table as a premise sentence and applies PLMs to encode both the table and statements into distributed representation for classification. This model excels at linguistic reasoning like paraphrasing and inference but lacks symbolic reasoning skills. Intuitively, encoding more table contents, e.g., type information and content snapshots, relevant to the input utterance could potentially help answer questions that involve reasoning over information across multiple rows in the table because they can provide more hints about the meaning of a column. TaPas (Herzig et al., 2020) is another newly proposed pretrained question answering model over tables implemented on BERT to avoid generating the logical forms. The model can fine-tune on semantic parsing datasets, only using weak supervision, with an end-to-end differentiable recipe.

Another stream of work on evidence finding with table is the rule-based approaches. Most evidence cells can be extracted by rules. For example, if a row head or column head appears in the statement, we infer this row or col support this statement. Although rule-based approaches suffer from the low recall issue, they exhibit high precision and can be applied to adjust the result for ensemble.

## 3 System Overview

We elaborate the task and present our system in the following.

### 3.1 Data Description and Tasks

In Task A, the original dataset is a set of XML files, where each XML file represents a table and contains multiple sections:

- *document* section represents the whole document;
- *table* section determines the unique ID of the document;
- *caption* section is a brief description of the table;
- *legend* section is a detailed description of the table;
- *multiple row* sections describe the contents of each row of the table; and
- *statements* section provides several factual statements.

This task aims to determine if a statement is entailed or refuted by the given table, or whether, as is in some cases, this cannot be determined from

		Tables	Label Distribution	Tokens in Statements	Tokens in Tables
			Entailed/Refuted/Unknown	Max./Min./Avg.	Max./Min./Avg.
Train	Task 9	981	2,818/1,688/0	88/3/11	302/1/10
	Tabfact	16,573	63,962/54,313/0	57/4/14	127/5/13
	Augm.	17,554	66,780/56,001/61,436	88/3/12	302/1/11
Dev.	—	52	250/213/93	53/4/13	115/2/11
Test	—	52	—	82/4/12	69/2/11

Table 1: Data statistics.

the table. The competition also provides two kinds of evaluations for the task:

- 3-way F1 score evaluation: a standard precision/recall evaluation (3-way) is computed to evaluate whether each table is correctly classified into one of the three types in {Entailed, Refuted, Unknown}. It is to test whether the classification algorithm understands cases where there is insufficient information to make a determination.
- 2-way F1 score evaluation: the F1 score is computed to evaluate the performance when the statements with the “unknown” ground truth label are removed. The metric will also penalize misclassifying Refuted/Entailed statement as unknown.

In the evaluation, the score for all statements in each table is first averaged and then averaged across all tables to get the final F1 score.

In Task B, the raw dataset is a subset of task A, where unknown statements are excluded. The goal is to determine for each cell and each statement, if the *cell is within the minimum set of cells* needed to provide evidence for the statement “relevant” or “irrelevant”. For some statements, there may be multiple minimal sets of cells that can be used to determine statement entailment or refusal. In such cases, the ground truth will contain all of the versions. The evaluation will calculate the recall and precision for each cell, with “relevant” cells as the positive category. The evaluation is conducted similarly as that in Task A.

### 3.2 Data Augmentation

There are mainly two critical issues in Task A. First, the number of the tables is small. We then include more external data, the TabFact dataset (Chen et al., 2020) to improve the generalization of our proposed system. Second and more critically, “unknown” statements do not exist in the training set but may appear in the test set. To allow our sys-

tem to output the “unknown” category, we construct additional “unknown” statements to enrich the training set. More specifically, we randomly select some statements from other tables and assign them to the “unknown” category for the current table. In order to keep balance on the labels, the number of selected statements from other tables is set to half of the statements in the current table. Details about the data statistics can be referred to Table 1.

### 3.3 Model Ensemble for Task A

Figure 1 outlines the overall structure of our system, which is an ensemble of two main pretrained models on table-based data, TaBERT and TaPas, or two variants of TaBERT and four variants of TaPas. It is worth noting that the input of all models are the same. That is, given a statement and a table, the input is started with the sentence token, [CLS], followed by the sequence of the tokens in the statement, the segmentation token ([SEP]), and the sequence of the tokens in the flattened table. All the tokens in the statement and the table are extracted by wordpiece as in BERT and related NLP tasks (Devlin et al., 2019; Yang, 2019; Yang et al., 2021; Yang and Shen, 2021; Wang et al., 2021b). The flattened table means that we borrow the implementation in TaBERT by only extracting the most likely content snapshot as detailed in Sec. 3.4. The obtained tokens’ embeddings are then fed into six strong baselines, i.e., two variants of TaBERT and four variants of TaPas, to attain the classification scores for the corresponding labels. The classification scores are then concatenated and fed into a vote layer, i.e., a fully-connected network, to yield the final prediction.

### 3.4 Content Snapshot

In order to pin point the important rows and avoid excessively encode input from the table, we borrow the idea of content snapshot in TaBERT (Yin et al.,

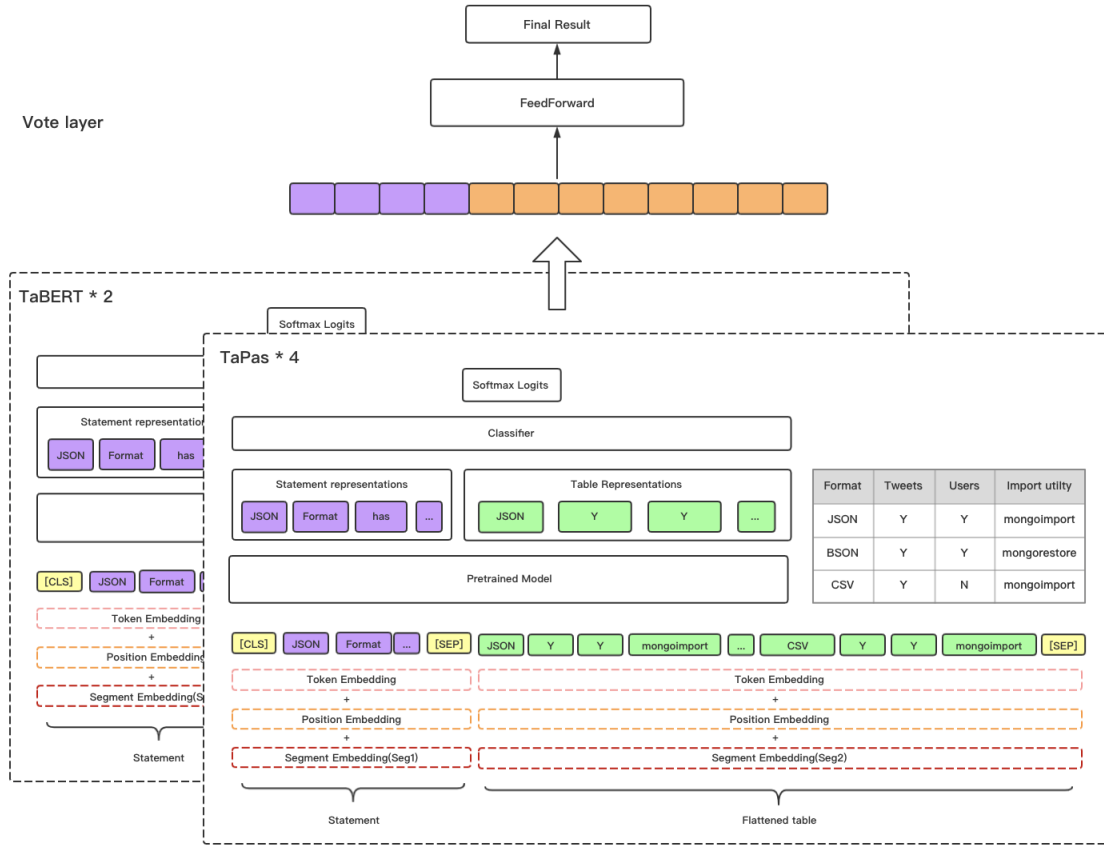


Figure 1: Ensemble architecture

Models	Original data	+TabFact	+Augm.
TaBERT_1	0.7446/0.6580	0.7634/0.6837	0.8003/0.7502
TaBERT_3	0.7689/0.6792	<b>0.7952/0.7008</b>	<b>0.8241/0.7653</b>
TaPas_TFIMLR	0.7502/0.6637	0.7859/0.6799	0.8102/0.7649
TaPas_WSIMLR	0.7498/0.6522	0.7852/0.7005	0.8024/0.7577
TaPas_IMLR	0.7538/0.6358	0.7799/0.6890	0.7908/0.7396
TaPas_WSMLR	<b>0.7695/0.6875</b>	0.7904/0.7058	0.8156/0.7609

Table 2: Comparison of strong baselines in Task A for 2-way and 3-way evaluation.

Models	+TabFact	+Augm.	+Rule
TaBERT_1	0.4025	0.4159	0.4605
TaBERT_3	0.4158	0.4305	0.4685
TaPas_TFIMLR	0.4253	0.4299	0.4597
TaPas_WSIMLR	0.4199	0.4208	0.4682
TaPas_IMLR	0.4006	0.4102	0.4467
TaPas_WSMLR	<b>0.4258</b>	<b>0.4386</b>	<b>0.4708</b>

Table 3: Comparison of strong baselines in Task B.

snapshot of  $K$  rows based on the following simple strategy. First, we count the number of rows of each table and find their median, say  $R$ . If the number of rows in the current table is less than or equal to  $R$ , then  $K$  is set to the total number of rows in the current table and the content snapshot is the entire content of the current table. If the number of rows in the current table is greater than  $R$ , we set  $K = R$  and select the top- $K$  row with the highest overlap rate between the statement and each row of  $n$ -grams as the candidate rows.

2020) to encode only a few rows that are most relevant to the statement. We create the content

### 3.5 Rule Construction for Task B

For Task B, we apply the same model trained in Task A to find whether the current table supports the statement. If yes, we label all cells as entailed. Otherwise, we first align the word expression in tables and statements while building the corresponding rules to adjust the model prediction. That is, we change uppercase to lowercase and transform all abbreviations into the full name in statements, cells, col heads and row heads. We also conduct stemming on all words. For example, “definition” and “defined” is transformed to “define”. After that, we collect all words in a statement into a word bag and determine the supporting relation based on the following rules: 1) If a word in the word bag appears in a row head, we then infer that cells in the whole column supports the statement; 2) If a word appears in the first column of the table, we then infer that cells in the whole row supports the statement; 3) If a word appears in both a row head and a cell in the first column of a table, we then infer that the cell corresponding to the row and column supports the statement; 4) If a word appears in a cell, we then infer that this cell supports the statement.

## 4 Experiments

In the following, we present the strong baselines and the results with analysis.

We have tried different combinations of TaBERT and TaPas pre-trained models and choose the following 6 best baselines: 1) TaBERT\_1: the pre-trained TaBERT with  $K = 1$ ; 2) TaBERT\_3: the pre-trained TaBERT with  $K = 3$ ; 3) TaPas\_TFIMLR: the pre-trained large TaPas downloaded from `tapas_tabfact_inter_masklm_large_reset.zip`; 4) TaPas\_WSIMLR: the pre-trained large TaPas downloaded from `tapas_wikisql_sqa_inter_masklm_large_reset.zip`; 5) TaPas\_IMLR: the pre-trained large TaPas downloaded from `tapas_inter_masklm_large_reset.zip`; 6) TaPas\_WSMLR: the pre-trained large TaPas downloaded from `tapas_wikisql_sqa_masklm_large_reset.zip`. Our proposed system is funetuned on the above models for the original training data, the original data with the TabFact data, and the augmentation data. We also tune the hyper parameters to fit a better result in the local test dataset.

Table 2 reports the evaluation results of Task

A on the development set when funetuning the above six strong baselines on different training data. The results show that the TaPas\_WSMLR attains the best performance on the original data. The best performance is further improved from 0.7695 to 0.7952 for 2-way evaluation and from 0.6875 to 0.7058 for 3-way evaluation, respectively, by including the TabFact data. The performance is further improved to 0.8241 for 2-way evaluation and 0.7653 for 3-way evaluation, respectively, by adding the augmentation data. Finally, we apply the voting mechanism to ensemble the results and achieve the F1 scores of 0.8496 and 0.7732 on the test set, respectively.

Table 3 reports the results of Task B on the development set when funetuning the above six strong baselines on different training data. The results show that the TaPas\_WSMLR attains the best performance among all six strong baselines and the perform increases from 0.4258 after adding the TabFact data, to 0.4386 after adding the augmentation data, and to 0.4708, additional 7.3% improvement after adding the manual rules. We conjecture that TaPas\_WSMLR can provide more complementary information for solving the task. Finally, we ensemble the results by the voting mechanism and achieve the F1 score of 0.4856 on the test set.

In sum, results in Table 2 and Table 3 confirm the effectiveness of our proposed system by including more training data and the manual rules.

## 5 Conclusion

In this paper, we present the implementation of our ensemble system to solve the problem of SemEval 2021 Task 9. To include more training data and resolve the issue of lacking data from the “Unknown” category in the training set, we include external corpus, the TabFact dataset, and specially construct the augmented data for the “Unknown” category. Content snapshot is also applied to reduce the encoding effort. Six pre-trained language models over tables are funetuned on the TabFact dataset and the augmented data with content snapshot tables to evaluate the corresponding performance. An ensemble mechanism is applied to get the final result. Moreover, data alignment and manual rule determination are applied to solve Task B. Finally, our system attains the F1 score of 0.8496 and 0.7732 in Task A for 2-way and 3-way evaluation, respectively, while getting the F1 score of 0.4856 in Task B.

## References

- Jfak Van Benthem. 2008. A brief history of natural logic. *London*, volume 55(2):339–346(8).
- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). *CoRR*, abs/1508.05326.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyong Zhou, and William Yang Wang. 2020. [Tabfact: A large-scale dataset for table-based fact verification](#). In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.
- D., C., P., Suppe, and Frederick. 1978. The structure of scientific theories. *The American Journal of Psychology*.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Andreas Hanselowski, Hao Zhang, Zile Li, Daniil Sorokin, Benjamin Schiller, Claudia Schulz, and Iryna Gurevych. 2018. [Ukp-athene: Multi-sentence textual entailment for claim verification](#). *CoRR*, abs/1809.01479.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. [Tapas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333. Association for Computational Linguistics.
- Wenqiang Lei, Yisong Miao, Rungpeng Xie, Bonnie Webber, Meichun Liu, Tat-Seng Chua, and Nancy F Chen. 2021. Have we solved the hard problem? it’s not easy! contextual lexical contrast as a means to probe neural coherence. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-sql. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954.
- Jiaqi Li, Ming Liu, Min-Yen Kan, Zihao Zheng, Zekun Wang, Wenqiang Lei, Ting Liu, and Bing Qin. 2020. Molweni: A challenge multiparty dialogues-based machine reading comprehension dataset with discourse structure. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2642–2652.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized BERT pretraining approach](#). *CoRR*, abs/1907.11692.
- Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.
- Alane Suhr, Stephanie Zhou, Iris Zhang, Huajun Bai, and Yoav Artzi. 2018. [A corpus for reasoning about natural language grounded in photographs](#). *CoRR*, abs/1811.00491.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021a. Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents (sem-tab-facts). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Xinyi Wang, Haiqin Yang, Liang Zhao, Yang Mo, and Jianping Shen. 2021b. Refbert: Compressing bert by referencing to pre-computed representations. In *IJCNN*.
- Haiqin Yang. 2019. [BERT meets chinese word segmentation](#). *CoRR*, abs/1909.09292.
- Haiqin Yang and Jianping Shen. 2021. Emotion dynamics modeling via bert. In *IJCNN*.
- Haiqin Yang, Xiaoyuan Yao, Yiqun Duan, Jianping Shen, Jie Zhong, and Kun Zhang. 2021. Progressive open-domain response generation with multiple controllable attributes. In *IJCAI*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. [Xlnet: Generalized autoregressive pretraining for language understanding](#). *CoRR*, abs/1906.08237.



Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. [Tabert: Pretraining for joint understanding of textual and tabular data](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. [TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data](#). *arXiv e-prints*, page arXiv:2005.08314.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2sql: Generating structured queries from natural language using reinforcement learning](#). *CoRR*, abs/1709.00103.

Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. [Retrieving and reading: A comprehensive survey on open-domain question answering](#). *arXiv preprint arXiv: 2101.00774*.

# Volta at SemEval-2021 Task 9: Statement Verification and Evidence Finding with Tables using TAPAS and Transfer Learning

Devansh Gautam\* Kshitij Gupta\* Manish Shrivastava

International Institute of Information Technology Hyderabad

{devansh.gautam, kshitij.gupta}@research.iiit.ac.in,  
m.shrivastava@iiit.ac.in

## Abstract

Tables are widely used in various kinds of documents to present information concisely. Understanding tables is a challenging problem that requires an understanding of language and table structure, along with numerical and logical reasoning. In this paper, we present our systems to solve Task 9 of SemEval-2021: *Statement Verification and Evidence Finding with Tables* (SEM-TAB-FACTS). The task consists of two subtasks: (A) Given a table and a statement, predicting whether the table supports the statement and (B) Predicting which cells in the table provide evidence for/against the statement. We fine-tune TAPAS (a model which extends BERT’s architecture to capture tabular structure) for both the subtasks as it has shown state-of-the-art performance in various table understanding tasks. In subtask A, we evaluate how transfer learning and standardizing tables to have a single header row improves TAPAS’ performance. In subtask B, we evaluate how different fine-tuning strategies can improve TAPAS’ performance. Our systems achieve an F1 score of 67.34 in subtask A three-way classification, 72.89 in subtask A two-way classification, and 62.95 in subtask B.

## 1 Introduction

There has been extensive work on verifying if a given textual context supports a given statement. Even though tables are also widely used to convey information, especially in scientific texts, there has been comparatively less work on verifying if a given table supports a statement. To this end, SemEval 2021 Task 9 (Wang et al., 2021) focuses on statement verification and evidence finding for tables from scientific articles in the English language. The task is divided into two subtasks - A and B. The aim of subtask A is to classify whether a given

statement is entailed or refuted according to the given table and associated table metadata (such as captions and legends) or whether the statement’s truth is unknown as it cannot be determined from the table. The aim of subtask B is to classify each cell in the table as relevant or irrelevant in determining whether the statement is entailed or refuted from the tabular evidence (the truth value of the statement is also provided).

Our systems use TAPAS (Herzig et al., 2020) trained with intermediate pre-training (Eisenschlos et al., 2020) for both the subtasks. For subtask A, we fine-tune TAPAS after adding a three-way classification head on top for classifying the statement as entailed/refuted/unknown. We also evaluate how transfer learning and standardizing tables to have a single header row can improve TAPAS’ performance. Due to the similarity between subtask B and table question-answering (which involves cell selection or cell selection followed by aggregation), we use the TAPAS architecture previously used for table question-answering and fine-tune it to select the relevant cells. We also evaluate how different fine-tuning strategies can improve TAPAS’ performance on evidence finding.

Our systems achieve an F1-micro score of 67.34 in subtask A and 72.89 in subtask A if the unknown statements are not considered while calculating the metrics (however, classifying entailed/refuted statements as unknown is still penalized). Our submitted system achieves an F1 score of 62.95 in subtask B. During the post-evaluation phase, we modified our system and achieved an F1-score of 65.48 in subtask B.

The code for our systems is available at <https://github.com/devanshg27/sem-tab-fact>.

\*The authors have contributed equally.

	(1)	(2)	(3)	(4)
	English Language		English	
	Frequency	Percent	Frequency	Percent
AQA	241539	61.6	84742	55.7
WJEC	83219	21.2	39650	26.1
Pearson	37194	9.5	18815	12.4
OCR	30061	7.7	8818	5.8
Total	392015		152025	

**Caption:** Number of GCSE Full Course entries by Awarding Body (KS4 Results tables, 2014)

**Legend:** Note. Number of GCSE Full Course entries in the summer season of the academic year 2012-2013. AQA (The Assessment and Qualifications Alliance); WJEC (Welsh Joint Education Committee); OCR (Oxford, Cambridge and RSA Examinations); CCEA (Council for the Curriculum, Examinations and Assessment). We do not show the information of an additional awarding body that accounts for almost no entries.

<i>Entailed:</i>	1. The highest Frequency, not counting the Total, is 84742. 2. The highest English Percent is for AQA
<i>Refuted:</i>	1. The highest Percent value for OCR is 5.8 2. The lowest total is 392015
<i>Unknown:</i>	1. First, this is due to technical problems in providing Unique Candidate Numbers (UPN) for all candidates. 2. This is for four main reasons.

Figure 1: An example from the SEM-TAB-FACTS dataset: Table A1 From 10262.xml along with its caption and legend. Some example statements of each class associated with this table are also shown. The highlighted cells are the relevant cells for entailed statement 2.

## 2 Background

Verifying if the given textual evidence supports a given statement is a fundamental natural language processing problem. It has been extensively studied under different tasks such as RTE (Recognizing Textual Entailment) (Dagan et al., 2006), NLI (Natural Language Inference) (Bowman et al., 2015), FEVER (Fact Extraction and VERification) (Thorne et al., 2018). In recent years, large-scale pre-trained models (Devlin et al., 2019; Peters et al., 2018; Yang et al., 2019; Liu et al., 2019) have dominated these tasks and have achieved close-to-human performance. NLVR (Suhr et al., 2017) and NLVR2 (Suhr et al., 2019) focus on verifying a statement given an image as evidence. TABFACT (Chen et al., 2020) focuses on verifying a statement given a table from Wikipedia<sup>1</sup> as evidence.

Along with releasing TABFACT, Chen et al. (2020) also discuss two promising approaches for tabular fact verification, Latent Program Algorithm(LPA) and Table-BERT. LPA is a semantic parsing approach that parses statements into programs (logical forms) and executes the programs against the table to predict the entailment decision. Most of the current models (Zhong et al., 2020; Shi et al., 2020; Yang et al., 2020) for TABFACT are semantic parsing approaches similar to LPA. Table-BERT encodes the linearized tables and statements using BERT-based models and directly pre-

dicts the entailment decision. Zhang et al. (2020) inject table structural information into the mask of the self-attention layer of BERT-based models, which helps the model learn better table representations. TAPAS (Herzig et al., 2020) extends BERT’s architecture to capture the tabular structure, and it showed competitive performance on various table question answering datasets: SQA (Iyyer et al., 2017), WTQ (Pasupat and Liang, 2015) and WikiSQL (Zhong et al., 2017). Eisenschlos et al. (2020) add an intermediate pre-training step before the fine-tuning step to TAPAS and show that it achieves state-of-the-art results on TABFACT and SQA (Iyyer et al., 2017). Their model is still 8 points behind human performance on TABFACT since tabular fact verification involves table understanding and complex reasoning.

While TABFACT also focuses on fact verification using tables as evidence, it focuses on tables from Wikipedia, whereas SemEval-2021 Task 9 (SEM-TAB-FACTS) instead focuses on tables from scientific articles and has a subtask related to evidence finding. Also, TABFACT did not have a neutral/unknown class, which they left out because of low inter-worker agreement due to confusion with refuted class. Figure 1 shows an example of a table from the SEM-TAB-FACTS dataset and the labels for the two subtasks.

<sup>1</sup><https://www.wikipedia.org/>

	Train (Auto) Set	Train (Manual) Set	Dev Set
Total number of tables with <code>&lt;thead&gt;</code> tag	1977	980	52
Number of tables with correct header prediction	1855(93.83%)	918(93.67%)	51(98.08%)
Number of tables with header prediction error is $\leq 1$	1966(99.44%)	972(99.18%)	52(100%)

Table 1: Header Prediction Statistics

	(1)	(2)	(3)	(4)
	English Language	English Language	English	English
	Frequency	Percent	Frequency	Percent
AQA	241539	61.6	84742	55.7
WJEC	83219	21.2	39650	26.1
Pearson	37194	9.5	18815	12.4
OCR	30061	7.7	8818	5.8
Total	392015		152025	

(a) Converting multi-row/multi-column cells to single cells

	(1)	(2)	(3)	(4)
	English Language	English Language	English	English
	Frequency	Percent	Frequency	Percent
AQA	241539	61.6	84742	55.7
WJEC	83219	21.2	39650	26.1
Pearson	37194	9.5	18815	12.4
OCR	30061	7.7	8818	5.8
Total	392015		152025	

(b) Standardizing the header rows of the table with single cells

Figure 2: Pre-processing and header standardization applied to the table shown in Figure 1.

### 3 System Overview

In this section, we provide a general overview of our systems for the two subtasks. We use TAPAS for both subtasks.

#### 3.1 Subtask A: Statement Verification

**Pre-processing** Since TAPAS only works on tables with single cells (cells which do not span multiple columns/rows) only, we first convert the tables with multi-row/multi-column cells to tables with only single cells by duplicating the value of the cell in every single cell the multi-row/multi-column cell spans. An example of the pre-processing is shown in Figure 2a.

**Header Standardization** We experiment with standardizing the pre-processed tables with multi-row headers to tables with a single header row since TAPAS was pre-trained on single header tables and TABFACT (which we want to use for transfer learning) also contains single header tables. We first predict the number of header rows using the following rules:

1. In many pre-processed tables, we found that the left-most column contained row names,

and either (a) all the header cells in the left-most column were empty, or (b) the cell value at the top-left corner was repeated in all the header cells below it, or (c) the cell at the top-left corner was not empty, but the header cells below it were empty. Based on these cases, we initially estimate the number of header rows as the number of rows at the top, such that all cells in the left-most column in those rows are either empty or have the same value as the cell at the top-left corner.

2. We also found that in many cases, there were multi-column cells in the header, which had more specific sub-headers in the rows below. To handle these cases, we increment the estimate of header rows until no two adjacent columns have the same header cell values.

We merge the predicted header rows into a single row by joining each column’s header cell values into a single cell with a newline as a separator. An example of header standardization is shown in Figure 2b. We were provided with HTML versions of the tables in the training and development set. We compare our predictions against the `<thead>` tags in the HTML tables to analyze our header prediction system’s performance. The results are shown in Table 1. We also find that in almost all of the cases, the predictions are either correct or have an error of  $\pm 1$ .

To study the effect of header standardization, we will train all our systems with and without header standardization.

**Model** Our model takes the following input: `[CLS] <statement> [SEP] <flattened table>`, which is tokenized using the standard BERT tokenizer. We compute the class probabilities using a linear layer with a softmax activation function on top of the output of the `[CLS]` token, as shown in Figure 3a. We use the weighted cross-entropy loss, which helps in handling imbalance in the class sizes:

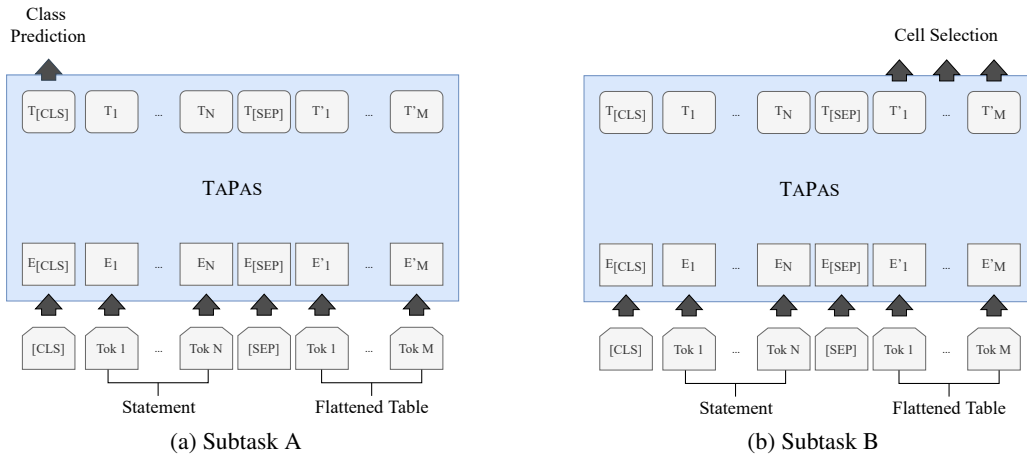


Figure 3: The architecture of our models

$$H_y(y') = - \sum_i \sum_{k=1}^K w_k y_{ik} \cdot \log(y'_{ik})$$

Where  $y_{ik}$  denotes the ground truth label, it is 1 if  $k$  is the true class label of the  $i^{th}$  token, and 0 otherwise,  $y'_{ik}$  is the corresponding model probability prediction and  $w_k$  is the weight for class  $k$ . We set  $w_k$  as the size of the biggest class divided by the size of class  $k$ .

To analyze how transfer learning can improve performance, we compare the following approaches:

- **TAPAS-stf:** We use the publicly available TAPAS checkpoint which has been pre-trained with a masked language modeling objective and fine-tune it on the SEM-TAB-FACTS dataset provided by the task organizers.
- **TAPAS-tf:** As a baseline, we directly use the publicly available TAPAS checkpoint, which had been fine-tuned on TABFACT without any further fine-tuning on SEM-TAB-FACTS. Since TABFACT has only entailed/refuted labels, this model is a binary classifier and does not predict the unknown class’s probabilities.
- **TAPAS-tf-stf:** We use the publicly available TAPAS checkpoint, which had been fine-tuned on TABFACT and further fine-tune it on the SEM-TAB-FACTS dataset released by the task organizers. This is our submitted model for subtask A.

### 3.2 Subtask B: Evidence Finding

#### Pre-processing and Header Standardization

We convert the multi-row/multi-column cells and standardize the header rows as discussed in Section 3.1. The relevant/irrelevant labels of the multi-row/multi-column cells are duplicated to all the single cells they span. We consider the relevant/irrelevant labels only for the cells of the non-header rows as TAPAS does not make predictions for header cells. Based on the performance of header standardization in subtask A (which we will discuss in Section 5), we standardize headers for all our models in this subtask.

**Model** Our model takes the following input:

[CLS] <statement> [SEP] <flattened table>, which is tokenized using the standard BERT tokenizer. We show the architecture of our model in Figure 3b. Our model computes token-level logits using a linear layer on top of each token’s last hidden state output, which are used to compute cell-level logits by averaging the logits of the tokens in each cell. The probability of selection for each cell is calculated from the cell-level logits using the sigmoid function. We use the weighted binary cross-entropy loss which helps in handling class imbalance:

$$H_y(y') = - \sum_i w_p y_i \cdot \log y'_i + (1 - y_i) \cdot \log (1 - y'_i)$$

Where  $y_i$  denotes the ground-truth label, it is 1 if the  $i^{th}$  token is part of any relevant cell, and 0 otherwise,  $y'_i$  is the corresponding model probability prediction, and  $w_p$  denotes the weight of the positive (relevant) class. We set  $w_p$  to 10.

	#Tables	#Entailed statements	#Refuted statements	#Unknown statements
<b>Train (Auto-generated)</b>	1980	92136	87209	0
<b>Train (Manually annotated)</b>	981	2818	1688	0
<b>Train (with unknown statements)</b>	981	2818	1688	4506
<b>Validation</b>	52	250	213	93
<b>Test</b>	52	274	248	131

(a) Subtask A

	#Tables	#Entailed statements	#Refuted statements	#Relevant cells	#Irrelevant cells
<b>Train (auto-generated)</b>	1980	92136	87209	1039058	15467957
<b>Validation</b>	51	233	191	3048	28495
<b>Test</b>	52	251	219	3458	26724

(b) Subtask B

Table 2: Dataset Statistics for each subtask

Due to the similarity of evidence finding with table question-answering, we use the publicly available TAPAS checkpoint, which was fine-tuned in a chain on SQA, WikiSQL, and finally WTQ. We compare the following fine-tuning strategies:

- **WTQ-base:** As a baseline, we fine-tune our model directly for relevant cell selection on SEM-TAB-FACTS.
- **WTQ-statement:** We again fine-tune the model for relevant cell selection on SEM-TAB-FACTS, but we try to include the information on whether the statement was entailed/refuted by modelling the statement as ‘Which cells entail “<statement>”?’ or ‘Which cells refute “<statement>”?’ . <statement> denotes the original statement.
- **WTQ-separate:** We fine-tune two separate models, one which predicts the relevant cells for entailed statements and another one for refuted statements. This is our submitted system for subtask B.

During the post-evaluation phase, we experimented with the publicly available TAPAS checkpoint, which was fine-tuned on TABFACT. Similar to the systems described above, we compare three systems based on this checkpoint: TABFACT-base, TABFACT-statement, and TABFACT-separate.

**Post-Processing** We further apply post-processing steps to obtain the final prediction from the cell classification. To predict the header’s relevant cells, we select the header cells for any column with cells selected as a relevant cell. We label multi-row/multi-column cells as relevant if

any of the single cells they span are predicted as relevant.

## 4 Experimental Setup

### 4.1 Data Description

We used the dataset provided by the task organizers for both subtasks. We did not use the table metadata in our systems.

For subtask A, dataset statistics and the official splits are shown in Table 2a. The provided training sets do not have any statements of the unknown class. So, we used the manually annotated training set to create a training set with unknown statements. Each statement of the manually annotated training set was added as an unknown statement to a different table chosen randomly. We used this dataset for training all our models for subtask A.

For subtask B, dataset statistics and the official splits are shown in Table 2b. We use the auto-generated training set for training all our models in subtask B.

### 4.2 Implementation

For the implementation of our systems, we used the HuggingFace Transformers<sup>2</sup> library (Wolf et al., 2020) and we used the AdamW optimizer available in PyTorch<sup>3</sup> (Paszke et al., 2019) with the default parameters (learning rates are specified below). All models were fine-tuned using a single Nvidia GeForce RTX 2080 Ti GPU.

We used the base variant of TAPAS, which has a hidden dimension of 768 in all our models. All the

<sup>2</sup> 🐼 Transformers, v4.2.0, <https://huggingface.co/transformers/>

<sup>3</sup> PyTorch, v1.7.1, <https://pytorch.org/>

F1 Score	Validation Set			Test Set		
	TAPAS-stf	TAPAS-tf	TAPAS-tf-stf	TAPAS-stf	TAPAS-tf	TAPAS-tf-stf
<i>Without header standardization</i>						
2-way micro	<b>72.1</b> $\pm 0.43$	69.42	71.01 $\pm 0.99$	68.01 $\pm 0.28$	70.97	<b>72.97</b> $\pm 1.37$
3-way micro	<b>66.41</b> $\pm 0.48$	58.97	65.76 $\pm 0.37$	61.59 $\pm 0.02$	57	<b>65.15</b> $\pm 0.81$
Refuted	67.95 $\pm 0.98$	64.31	<b>70.32</b> $\pm 0.91$	62.04 $\pm 0.45$	64.05	<b>69.13</b> $\pm 0.74$
Entailed	67.8 $\pm 0.36$	58.94	<b>68.09</b> $\pm 1.24$	64.89 $\pm 0.49$	61.9	<b>67.23</b> $\pm 1.18$
Unknown	<b>49.76</b> $\pm 0.73$	0	47.52 $\pm 3.52$	<b>47.58</b> $\pm 0.8$	0	46.43 $\pm 1.88$
<i>With header standardization</i>						
2-way micro	71.34 $\pm 0.96$	72.78	<b>74.35</b> $\pm 1.14$	68.67 $\pm 0.9$	73.79	<b>73.87</b> $\pm 0.87$
3-way micro	66.16 $\pm 0.64$	61.11	<b>69.16</b> $\pm 0.58$	61.99 $\pm 0.8$	59.32	<b>66.95</b> $\pm 0.27$
Refuted	68.22 $\pm 0.29$	65.98	<b>73.2</b> $\pm 0.83$	61.42 $\pm 1.9$	65.7	<b>70.39</b> $\pm 0.44$
Entailed	67.98 $\pm 0.43$	63.67	<b>70</b> $\pm 1.69$	65.67 $\pm 0.21$	65.38	<b>68.9</b> $\pm 0.48$
Unknown	49.9 $\pm 3.07$	0	<b>50.91</b> $\pm 3.99$	48.27 $\pm 1.55$	0	<b>50.89</b> $\pm 3.93$

Table 3: Performance on subtask A: Mean and standard deviation of the metrics from 3 independent runs. In the case of TAPAS-tf, we calculate the metrics using the publicly available TAPAS checkpoint fine-tuned on TABFACT.

Model	Validation Set			Test Set		
	F1	F1 <sub>entailed</sub>	F1 <sub>refuted</sub>	F1	F1 <sub>entailed</sub>	F1 <sub>refuted</sub>
WTQ-base	55.39 $\pm 0.53$	64.07 $\pm 0.65$	48.66 $\pm 0.47$	61.36 $\pm 1.47$	68.47 $\pm 2.49$	<b>52.75</b> $\pm 1.15$
WTQ-statement	55.18 $\pm 1.78$	63.36 $\pm 3.16$	48.45 $\pm 0.8$	58.93 $\pm 2.49$	65.22 $\pm 4.38$	51.27 $\pm 0.54$
WTQ-separate	<b>56.46</b> $\pm 0.43$	<b>66.91</b> $\pm 0.3$	<b>48.74</b> $\pm 1.01$	<b>62.26</b> $\pm 0.79$	<b>71.87</b> $\pm 1.2$	50.79 $\pm 1.86$
<i>During Post-Evaluation Phase</i>						
TABFACT-base	58.41 $\pm 0.84$	64.88 $\pm 1.37$	54.02 $\pm 0.91$	61.46 $\pm 0.33$	67.32 $\pm 1.01$	54.47 $\pm 0.55$
TABFACT-statement	58.92 $\pm 1.69$	65.41 $\pm 1.95$	<b>54.18</b> $\pm 1.69$	62.78 $\pm 1.71$	68.44 $\pm 2.34$	<b>55.8</b> $\pm 1.36$
TABFACT-separate	<b>59.47</b> $\pm 0.23$	<b>68.06</b> $\pm 0.79$	53.16 $\pm 1.18$	<b>65.01</b> $\pm 0.6$	<b>74.18</b> $\pm 0.6$	54.48 $\pm 0.58$

Table 4: Performance on subtask B: Mean and standard deviation of the metrics from 3 independent runs

TAPAS checkpoints we used had been trained with intermediate pre-training and used relative position embeddings (the position index reset when a new cell starts).

For subtask A, we first fine-tuned the classifier head with the TAPAS layers frozen for 3 epochs with a learning rate of  $1^{-5}$  and then fine-tuned the whole model for 10 epochs with a learning rate of  $1^{-6}$ . We used a batch size of 8. We saved a checkpoint every 100 steps and selected the best checkpoint based on the validation set performance.

For subtask B, we fine-tuned the whole model for 5000 steps with a learning rate of  $1^{-6}$ . We used a batch size of 8. We saved a checkpoint every 50 steps and selected the best checkpoint based on the validation set performance.

### 4.3 Evaluation Metrics

In subtask A, two evaluation metrics are used. The first evaluation metric used is the standard F1-micro score for three-way classification. The second metric again calculates the F1-micro score but does not consider statements with their ground truth label as the unknown class for evaluation; however, classifying the entailed/refuted statements as unknown is penalized.

In subtask B, the evaluation metric used is the standard F1 score with relevant cells as the positive class. If multiple minimal sets of cells can be used to determine the statement’s truth value, the dataset contains all of these versions. The score for that statement is calculated by comparing the prediction against each ground truth version and considering the highest score.

	Validation Set		Test Set	
	Length( $\leq 512$ )	Length( $> 512$ )	Length( $\leq 512$ )	Length( $> 512$ )
<i>Distribution - Number of samples</i>				
Subtask A	431(77.52%)	125(22.48%)	616(94.33%)	37(5.67%)
Subtask B	345(81.37%)	79(18.63%)	442(94.04%)	28(5.96%)
<i>Performance of each task’s best model</i>				
Subtask A 2-way F1-micro	77.83 $\pm 0.57$	65.49 $\pm 3.13$	73.83 $\pm 0.83$	74.44 $\pm 1.57$
Subtask A 3-way F1-micro	73.13 $\pm 1.13$	55.53 $\pm 1.4$	66.71 $\pm 0.35$	54.74 $\pm 1.12$
Subtask B F1	62.79 $\pm 0.39$	45.91 $\pm 0.68$	65.38 $\pm 0.63$	58.98 $\pm 1.22$

Table 5: Results on long sequences

## 5 Results

**Subtask A** The performance of the various systems we considered in subtask A is shown in Table 3. Header standardization improves the performance of all the systems we compared. Transfer learning from TABFACT also improves the performance of our systems. Surprisingly, TAPAS-tf without any fine-tuning on SEM-TAB-FACTS has a better two-way F1-micro score than TAPAS-stf. This shows us the potential of transfer learning from TABFACT in subtask A.

From the confusion matrix shown in Figure 4a, we observe that our model struggles with the unknown class and often misclassifies it as refuted.

**Subtask B** The performance of the various systems we considered in subtask A is shown in Table 4. Modifying the statement to include entailed/refuted class information leads to a small drop in performance for the models fine-tuned on question-answering earlier and led to a small increase in performance in models fine-tuned on TABFACT. Separate models for entailed/refuted statements perform the best among the systems we considered. It significantly improves the performance on entailed statements, with a little drop in performance on refuted statements. Surprisingly, we observe that transfer learning from TABFACT performs better than transfer learning from WTQ, even though it is a cell selection task. We believe this is because the model has to predict the cells that can be used as evidence for table entailment. The token-level embeddings of the model fine-tuned on TABFACT are better for this task than the model fine-tuned on WTQ, which is instead a question-answering dataset.

Predicted Class	R	186 75%	72 26%	21 16%
	E	50 20%	195 71%	52 40%
	U	12 5%	7 3%	58 44%
		R	E	U
		Target Class		

(a) Subtask A

Predicted Class	I	24596 92%	903 26%
	R	2128 8%	2555 74%
		I	R
		Target Class	

(b) Subtask B

Figure 4: Confusion matrices of the test set predictions by our best model for each subtask. The percentages show the ratio of the target class, which was predicted as that class.

**Long Inputs** The maximum number of tokens supported by our system is 512. In sequences longer than 512 tokens, the tables are truncated row by row to fit in 512 tokens. We compare our system’s performance on these long sequences and sequences that fit within 512 tokens. The results are shown in Table 5. We find a significant drop in performance on sequences longer than 512 tokens which had to be truncated.

## 6 Conclusion

In this paper, we presented our approach for fact verification and evidence finding for tabular data in scientific documents. We show that transfer learning from TABFACT and standardization of the tables to have a single header helps improve our system’s performance. We also show that having separate evidence finding models for entailed/refuted statements helps improve our system’s performance in the second subtask.

We also find that our model has a significant



drop in performance on large tables since they are truncated to fit in the 512 tokens, the maximum number of tokens supported by TAPAS.

In future work, we would like to experiment with table pruning methods like Heuristic entity linking (Chen et al., 2020) or Heuristic exact match (Eisenschlos et al., 2020) so that the statement and table can fit in 512 tokens. Our systems did not use the table metadata while making the predictions. In the future, we would also like to explore extending the model to encode table metadata along with the table.

## Acknowledgments

We thank the organisers of the shared task for their effort, and the anonymous reviewers for their insightful comments.

## References

- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyu Zhou, and William Yang Wang. 2020. [TabFact: A large-scale dataset for table-based fact verification](#). In *International Conference on Learning Representations*.
- Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. [Understanding tables with intermediate pre-training](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.
- Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. [Search-based neural structured learning for sequential question answering](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [RoBERTa: A robustly optimized bert pretraining approach](#).
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. [PyTorch: An imperative style, high-performance deep learning library](#). In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.
- Qi Shi, Yu Zhang, Qingyu Yin, and Ting Liu. 2020. [Learn to combine linguistic and symbolic information for table-based fact verification](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5335–5346, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. [A corpus of natural language for visual rea-](#)

- soning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, Vancouver, Canada. Association for Computational Linguistics.
- Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. 2019. [A corpus for reasoning about natural language grounded in photographs](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy. Association for Computational Linguistics.
- James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. [FEVER: a large-scale dataset for fact extraction and VERification](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. [SemEval-2021 Task 9: A fact verification and evidence finding dataset for tabular data in scientific documents \(SEM-TAB-FACTS\)](#). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. [Transformers: State-of-the-art natural language processing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.
- Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. [Program enhanced fact verification with verbalization and graph attention network](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. [XLNet: Generalized autoregressive pretraining for language understanding](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. [Table fact verification with structure-aware transformer](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics.
- Victor Zhong, Caiming Xiong, and Richard Socher. 2017. [Seq2SQL: Generating structured queries from natural language using reinforcement learning](#).
- Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. [Logical-FactChecker: Leveraging logical operations for fact checking with graph module network](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

# Kaushik Acharya at SemEval-2021 Task 9: Candidate Generation for Fact Verification over Tables

Kaushik Acharya

Philips India Ltd. / Bangalore, India

acharya.kaushik@gmail.com

## Abstract

This paper describes the system submitted in the SemEval-2021 Statement Verification and Evidence Finding with Tables task. The system relies on candidate generation for logical forms on the table based on keyword matching and dependency parsing on the claim statements.

## 1 Introduction

Tables convey important information in a concise manner. This is true in many domains, scientific documents being one of them. Truth verification tasks in past (e.g. SemEval-2019 Fact Checking Task) have focused on written text without considering the tables. The current shared task (Wang et al., 2021) focuses on tables written in English language. It requires participants to develop systems to predict

- veracity of textual claims (statement verification)
- identify table cells forming relevant evidence for the claim (evidence finding)

The shared task <sup>1</sup> comprised of two sub-tasks:

1. Subtask A: **Table Statement Support**
2. Subtask B: **Relevant Cell Selection**

*Subtask A* is a classification problem in which the system needs to assign one of the following labels for the claim statement:

- *Entailed*: Table supports the statement.
- *Refuted*: Statement is contradicted by the table.
- *Unknown*: Not enough information available in the table to assess statement’s veracity.

Country	N	Board network			Family network			State network			Political network		
		mean	SD	max	mean	SD	max	mean	SD	max	mean	SD	max
Hong Kong	133	5.12	6.1	33	2.62	4.51	26	1.00	1.41	6	0.67	1.37	6
Indonesia	169	1.64	3.31	23	0.95	2.64	17	0.14	0.38	2	0.22	1.09	9
Japan	126	1.84	2.33	15	0.07	0.42	3	0.09	0.31	2	0.00	0.00	0
South Korea	133	2.5	2.8	21	1.09	1.37	6	0.15	0.40	2	0.02	0.15	1
Malaysia	281	7.35	6.61	37	1.07	1.94	8	2.15	3.09	18	0.36	0.74	5
Philippines	98	8.52	8.91	38	5.33	6.16	21	0.71	1.59	10	0.20	0.81	6
Singapore	116	3.52	3.24	15	0.59	1.66	12	1.28	2.40	11	0.57	1.90	14
Taiwan	107	1.6	2.22	12	0.21	1.11	7	0.14	0.46	3	0.00	0.00	0
Thailand	127	5.11	5.04	23	1.58	3.15	19	0.73	1.99	11	0.29	1.16	8

Figure 1: Example table showing Networks across East Asia.

id	Claim statement
1	The n value is same for Hong Kong and Malaysia.
2	There are 9 different types country in the given table.

Table 1: Statement claims of table in Figure 1

*Subtask B* requires finding evidence (table cells) which are minimally required to either support or refute the claim statement. This is not applicable for the statements whose veracity is unknown.

These tables were sourced from scientific articles belonging to journals published by Elsevier and available on ScienceDirect. For details related to web scraping of the articles, selection criteria for choosing the tables, creating statement claims and assigning table cell evidence for the claim, please refer to the task description paper (Wang et al., 2021). An example table is shown in Fig. 1 with corresponding claims mentioned in Table 1.

System described in this paper generates logical form candidates on the table data frame based on the claim statement. It executes the most probable candidate and verifies the output to check whether it matches with the one mentioned in the statement. Averaged F1 score over the tables are shown in

<sup>1</sup><https://competitions.codalab.org/competitions/27748>

Table 4. Source code has been released on github <sup>2</sup>.

## 2 Related Work

Thorne et al. (2018) had conducted Fact Extraction and VERification (**FEVER**) Shared Task<sup>3</sup> to build systems to verify claims based on evidence from Wikipedia. Similar to the current shared task, systems had to label claim as **Supported**, **Refuted** or **NotEnoughInfo** (if there isn't sufficient evidence to either support or refute it). Along with that, system must extract textual evidence (sets of sentences) that support or refute the claim.

Pasupat and Liang (2015) had created question answering dataset (WikiTableQuestions) <sup>4</sup> from Wikipedia tables. Using question-answer pairs as supervision, they developed a logical-form driven parsing algorithm.

Herzig et al. (2020) build a question answering model over tables without generating logical forms by extending BERT's architecture (Devlin et al., 2019) with additional positional embeddings to encode tabular structure.

## 3 Model Description

### 3.1 XML Data

Input xml documents were parsed using the *ElementTree XML API* <sup>5</sup>. Each document is composed of table element(s). Table element is composed of row elements and statements. The row elements describe the rows and columns of the table and contains the text of each table cell. Optionally, legend and caption texts were also provided for a portion of tables.

### 3.2 Loading table

Table element was parsed and loaded into *pandas dataframe* <sup>6</sup>. The challenging part was in identifying column labels having hierarchical indexing <sup>7</sup>. An example of hierarchical columns is shown in Fig. 1, which has four columns: Broad network,

<sup>2</sup>[https://github.com/kaushikacharya/statement\\_verification\\_evidence\\_finding](https://github.com/kaushikacharya/statement_verification_evidence_finding)

<sup>3</sup><https://fever.ai/2018/task.html>

<sup>4</sup><https://ppasupat.github.io/WikiTableQuestions/>

<sup>5</sup><https://docs.python.org/3/library/xml.etree.elementtree.html>

<sup>6</sup><https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

<sup>7</sup>[https://pandas.pydata.org/pandas-docs/stable/user\\_guide/advanced.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/advanced.html)

#	Candidate
1	Column superlative value
2	Column values identical or unique
3	Comparison of column values for a row or vice-versa
4	Column value range

Table 2: Partial list of Candidates

Family network and so on. All these columns have three sub-columns: mean, SD, max. A simple approach to identify whether multiple table rows represent column labels was applied: Table rows from top were considered as part of column labels until all the columns of the table are filled. In the example table, *Broad network* is mentioned in table cell col:2 and *Family network* belongs to col:5. The in-between columns (i.e. 3,4) were filled in next row. Hence the first two rows were considered as column labels.

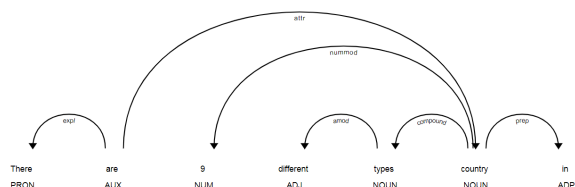


Figure 2: Dependency tree for statement id=2 in Table 1

### 3.3 Candidate Selection

A list of candidates were defined along with pattern rules to identify them and corresponding operations to execute. For instance, for the candidate superlative(highest/lowest) of column, corresponding operation on pandas dataframe gets executed. Statements are parsed to identify the candidate and fact verification is done in the following steps:

1. Match column and row labels (if available) based on approximate keyword matching.
2. Candidate operation(s) are identified based on keyword and dependency tag matching.
3. Candidate logical form is executed.
4. Output of candidate operation is matched with the one mentioned in the statement.

Table 2 enumerates subset of candidates.

For matching columns and row labels, sliding window over the tokens of statements are matched using Jaro metric (Cohen et al., 2003). Number

of tokens in column/row label is considered as the window size. As an example, for matching the row label *Hong Kong* of Figure 1 in Table 1, sliding window of two tokens is considered. If overlapping window match the same column/row label, the one with maximum Jaro metric is chosen. A span of statement tokens is considered matched if the Jaro metric is above a threshold (value considered 0.85).

**Examples:** Statement id=1 in Table 1 matches the candidate: *comparison of two rows for a column value*. The two rows referred by *Hong Kong* and *Malaysia* are compared for the value corresponding to the column *N*. Logical form: Comparing whether the cell value for the matched column corresponding the matched rows is same/different.

Statement id=2 refers to the candidate: *unique count of the values under the column Country*. Candidate is chosen based on the keyword *different* and matching of a single column **Country**. Candidate value (that needs to be verified) is identified based on the dependency tree shown in Figure 2. The token 9 with part of speech tag *NUM* has the head token of column *country* through dependency tag *nummod*. The logical form that has been assigned for the candidate is unique count over the matched column of the *pandas dataframe*.

## 4 Experimental Setup

### 4.1 Data

The data splits used were the same as provided by the task organizers. Split statistics is shown in Table 3.

split	docs	tables	claims
train	424	981	4506
dev	21	52	556
test	36	52	653

Table 3: Data split count statistics

### 4.2 External libraries

- spaCy (version: 2.3.2) <sup>8</sup>
- word2number <sup>9</sup>

### 4.3 Evaluation Metrics

**Task A - Fact Verification** The goal of this task is to determine whether a statement can be en-

<sup>8</sup><https://github.com/explosion/spaCy>

<sup>9</sup><https://github.com/akshaynagpal/w2n>

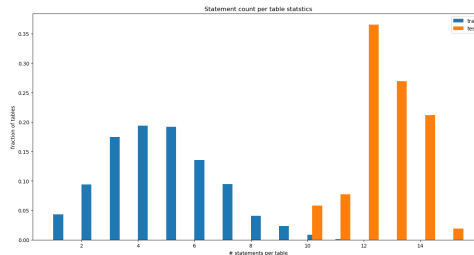


Figure 3: Distribution of number of statements per table.

tailed/refuted by the given table, or cannot be determined from the table. The classification algorithm is evaluated using the standard F1-score. Two different evaluation results were generated:

1. Two way
2. Three way

*Two way* is an easier evaluation in which **unknown** ground truth labels were ignored. Whereas in *three way* all the three labels were considered. This tests whether the classification algorithm understands cases where there are insufficient information to make prediction.

**Task B - Evidence Finding** The goal of this task is to determine whether a table cell is needed to entail/refute the given statement. In other words, whether a statement can be entailed/refuted given only the table cells marked as *relevant*. F1 score is computed for each table with *relevant* cells as the positive class and *irrelevant* cells as the negative class.

Fig. 3 shows that unlike test data, train data has many tables with very few statements. This indicates that comparing averaged F1 score (as shown in Table 4) between train and test data is not a good indicator how well the model works on unseen test data compared to its performance on train/dev data. Instead comparing confusion matrix (as shown in Table 5 and Table 6) is a better indicator.

## 5 Results

Averaged F1 score over the tables are shown in Table 4. These are the scores at the time of writing this paper. Train data didn't had the ground truth for the relevant cell selection. Hence value is non-available for Task B on train data. Due to absence of *unknown* class in train data, 2-way and 3-way averaged F1 scores are same. Confusion matrix

is displayed in Table 5 for train data and Table 6 for test data. Predicted claim class *unknown* also contains the claim statements for which the system failed to identify candidate. Hence this class shows a high value.

split	Task A (2 way)	Task A (3 way)	Task B
train	0.3669	0.3669	NA
dev	0.3804	0.4314	0.3687
test	0.4037	0.4909	0.3849

Table 4: F1 score averaged over tables

truth	predicted		
	entailed	refuted	unknown
entailed	863	472	1483
refuted	49	861	778

Table 5: Confusion Matrix for Task A on train data. Row represents truth classes and column represents the predicted classes.

truth	predicted		
	entailed	refuted	unknown
entailed	83	39	152
refuted	8	127	113
unknown	3	15	113

Table 6: Confusion Matrix for Task A on test data. Rows and columns represents truth and predicted classes respectively.

type	# statements
Total wrongly predicted statements	2782
Neither column nor row matched	786
No column matched but row(s) matched	386

Table 7: Matching columns/rows in wrongly predicted statements in train data.

**Error Analysis:** The errors can be categorized broadly into the following categories:

1. Absence of semantic matching
2. Lack of enough candidate generation rules
3. Classifying candidate in a deterministic way

Due to keyword matching the system fails to identify the columns which are mentioned with

different words in the statement even though semantically they are same. Table 7 gives a glimpse of *probable* failures under this category.

The set of candidate generation rules needs to be extended. The current system misses out candidate generation in several statements because of the absence of these not yet defined candidates. The high number of *unknown* predictions in the confusion matrices shown in Table 5 and 6 is a proof of this issue. There’s a need for a scoring system which considers multiple probable candidates for logical forms. The current system selects a single candidate based on the one which matches first in the order listed for candidate match.

## 6 Conclusion

I have described the system used for submission to the Statement Verification and Evidence Finding with Tables task. The problem has been framed as a candidate generation for logical forms over dataframe using keyword matching and dependency parsing. Future work would include extending the defined list of candidates and usage of scoring based system to identify the most probable candidate. This improvement would take ideas from (Pasupat and Liang, 2015) for feature extraction and build a log-linear model to compute score for the candidates.

## References

- William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web, IIWEB’03*, page 73–78. AAAI Press.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of deep bidirectional transformers for language understanding*. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. *TaPas: Weakly supervised table parsing via pre-training*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Oana Cocarascu, Christos Christodoulopoulos, and Arpit Mittal. 2018. [The fact extraction and VERification \(FEVER\) shared task](#). In *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. [SemEval-2021 Task 9: Fact Verification and Evidence Finding for Tabular Data in Scientific Documents \(SEM-TAB-FACTS\)](#). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.

# AttesTable at SemEval-2021 Task 9: Extending Statement Verification with Tables for Unknown Class, and Semantic Evidence Finding

Harshit Varma and Aadish Jain

Indian Institute of Technology Bombay

Mumbai, Maharashtra, India

{harshitvarma, aadishjain}@cse.iitb.ac.in

Pratik Ratadiya and Abhishek Rathi

vCreaTek Consulting Services Pvt Ltd

Pune, Maharashtra, India

{pratik.r, abhishek.r}@vcreatek.com

## Abstract

This paper describes our approach for Task 9 of SemEval 2021: *Statement Verification and Evidence Finding with Tables*. We participated in both subtasks, namely statement verification and evidence finding. For the subtask of statement verification, we extend the TAPAS model to adapt to the ‘unknown’ class of statements by finetuning it on an augmented version of the task data. For the subtask of evidence finding, we finetune the DistilBERT model in a Siamese setting.

## 1 Introduction

Tables provide a compact and structured way of presenting information. They are easily interpretable by humans and are widely used in scientific papers, business articles, and even in government reports. At the same time, it is easy to misinterpret tabular data and even use it maliciously to spread misinformation. Thus, systems that can verify facts from tables and locate evidence in them can go a long way in ameliorating issues related to table interpretation. Such systems can also be used for question-answering over large tables, that are difficult to analyze manually, since the task of fact verification with evidence finding is closely related to that of question answering (Jobanputra, 2019).

Table entailment refers to the task of finding whether a sentence is refuted or supported by a given table. Traditionally, it has been considered a binary classification task. However, there could be instances where the table is not capable enough to either refute or accept the given statement, meaning the statement context is “unknown” for the table. In this paper, we have presented a table entailment setup that extends to three classes: refuted, entailed,

and unknown, as a part of the Sem-Tab-Fact task (Wang et al., 2021). The task could be divided into two parts: statement verification, and evidence finding. Given a table and a statement, one has to first determine whether the table entails the statement, refutes the statement, or has insufficient information about it. This forms the subtask of *statement verification*. If the table entails or refutes the statement, one would also like to know which cells in the table provided evidence for the same. This is the subtask of *evidence finding*, that can be formulated as a binary classification problem. Each cell in the table is assigned one of the ‘relevant’ or ‘irrelevant’ labels depending on whether it provided evidence for the given statement.

Recent years have seen a rise in the use of transfer learning in language processing (Malte and Ratadiya, 2019a) owing to their superior performance. Models based on underlying concepts like attention mechanism and transformers are seeing widespread use across a range of tasks (Malte and Ratadiya, 2019b; Ratadiya and Mishra, 2019). Our findings concurred with this trend as we used similar kinds of architectures as the fundamental blocks in the systems for both the subtasks. For the subtask of *statement verification*, we modify the recently released TAPAS model (Eisenschlos et al., 2020) that is pre-trained on the TabFact dataset (Chen et al., 2020). The pre-trained model is trained on only two classes: ‘entailed’ and ‘refuted’, and is not capable of classifying the ‘unknown’ samples. The training data provided to us for our task also contains the same two labels. To adapt to the ‘unknown’ statements, we augment the given data by including random statements from other tables as unknown. We then fine-tune the TAPAS model by extending it to three classes on this augmented data.



Using this approach, we achieve a three-way F1 score of 65.59 and a two-way F1 score of 71.72 on the test data. We ranked 8<sup>th</sup> on the official leaderboard for this subtask.

For the subtask of *evidence finding*, we fine-tune the DistilBERT model (Sanh et al., 2019) in a Siamese setup (Reimers and Gurevych, 2019). The model is pre-trained on the SNLI dataset (Bowman et al., 2015), the MultiNLI dataset (Williams et al., 2018) and the STS benchmark (Cer et al., 2017). For finetuning, we use the Contrastive loss (Hadsell et al., 2006) as our loss function. For making a prediction, we calculate the cosine similarity between the embeddings computed by the model for both the statement and the cell value. If the cosine similarity value crosses a set threshold, we consider the cell to be relevant to the given statement. Using this approach, we achieve an F1-score of 43.02 on the test data and secured the 7<sup>th</sup> rank on the official leaderboard for this subtask.

The source code of our proposed approach for both the subtasks has been made public<sup>1</sup> to encourage usage and improve the reproducibility of the results.

## 2 Related Work

Recently, the TabFact dataset was released, which contained 118K human-annotated statements, related to about 16K Wikipedia tables. These statements were annotated for only two labels, ‘entailed’ and ‘refuted’, leaving out the ‘unknown’ cases. A system’s ability to distinguish ‘unknown’ statements from ‘entailed’ and ‘refuted’ is quite critical, as one may elusively create seemingly believable statements that are actually ‘unknown’.

Recently, there has been work in areas related to table fact verification, especially since the release of the TabFact dataset. Many of these approaches are graph-based in nature (Shi et al., 2020; Yang et al., 2020; Zhong et al., 2020). The current state-of-the-art on the TabFact dataset is the recently released TAPAS model, which outperforms its predecessor by approximately 6%. Unfortunately, all of these models are quite far from human-level performance, suggesting ample scope for improvement.

Little to no work has been done previously on the task of evidence finding, with the closest approaches being related to table-based question

<sup>1</sup>The code is available at <https://www.github.com/vcreatek/attestable-semeval/>

answering like WikiTableQuestions (Pasupat and Liang, 2015), WikiSQL (Zhong et al., 2018) and SQA (Iyyer et al., 2017) where getting the answer for a question can be considered analogous to getting evidence for a statement.

## 3 System Overview

### 3.1 Data and Preprocessing

The training dataset for this task contains two kinds of data:

- **Manual annotations:** These are crowd-sourced from human annotators and have been validated by a second round of validation.
- **Auto-generated annotations:** These statements are auto-generated using a random paraphraser and table understanding service (Zheng et al., 2020).

A separate development set was also provided.

#### 3.1.1 Subtask A: Statement Verification

The provided dataset for this task is in XML format. We use a custom parser to convert the data into CSV format, such that each data sample is of the form (*table, statement, label*), where the table is also a CSV, extracted from the corresponding XML file. No preprocessing is applied to the statements.

In general, the table cells can span across multiple rows and columns. To simplify things, we treat a cell spanning multiple columns/rows as multiple cells with the same value. This preserves the logical hierarchy in the table while keeping the table structure simple. See Figure 1 for an example.

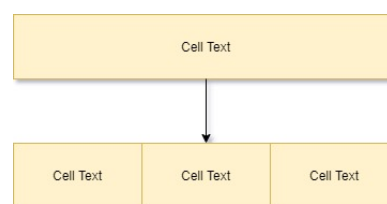


Figure 1: A cell spanning three columns is considered as three separate cells with the same value

A table may also have additional metadata accompanying it, like the legend, the caption, and the footer. Our final model does not use this metadata since we observe in Section 5.1.2 that including the metadata has a small negative impact on the model performance.

After the above preprocessing, we have 179,345 autogenerated and 4506 manual data samples.

### 3.1.2 Subtask B: Evidence Finding

For this subtask, we prepare the data in the form (*cell text, statement text, label*), where the label can be ‘relevant’ or ‘irrelevant’. We do not encode the table’s structural information and focus only on the semantic similarity between the statements and the cell texts.

An important point to note here is that only a few of the cells in the table actually contribute to the evidence for each statement. The average ratio of the number of relevant samples to total samples as observed in the development set for each statement is around 0.097. This causes a substantial class imbalance since a meager 7% of the total samples are ‘relevant’. To attend this problem, we undersample the ‘irrelevant’ samples by removing some irrelevant samples for each statement from the data. We try three ratios for undersampling and compare the percentage of relevant statements after undersampling in Table 1.

$n'_i$	% r	$n_t$
$0.6n_i$	10.15	10.24
$0.5(n_r + n_i)$	11.24	9.25
$0.4n_i$	14.54	7.15

Table 1:  $n_i$  and  $n_r$  stand for the number of irrelevant and relevant samples for a statement.  $n'_i$  stands for the no. of irrelevant samples kept for each statement after the undersampling. %r stands for the percentage of relevant samples after undersampling.  $n_t$  stands for the total samples after undersampling, in millions.

In the original data, the average ratio of relevant to total samples is around 9.7%, so we would want our data to reflect a similar ratio. Therefore, we use the data undersampled using the first approach for further analysis. Hence, the total number of samples after undersampling stand at 10.24M.

### 3.2 Data Augmentation

The given data has no ‘unknown’ statements; thus, we augment the data to introduce samples of this class label. Let  $S$  denote the set of all statements in our data. Let  $S_e$  denote the set of all entailed statements in our data. Let  $S_r$  denote the set of all refuted statements in our data. Then  $S = S_e \cup S_r$ .

For a given table  $t$ , let  $S_t$  denote the set of all statements associated with that table. We first calculate  $n_e$  (the number of entailed statements for that table) and  $n_r$  (the number of refuted statements for that table). We then calculate  $n_u$  (the number of

unknown statements to add) as follows:

$$n_u = \max(\min(n_e, n_r), 1) \quad (1)$$

After this, we randomly select  $n_u$  statements from  $S \setminus S_t$ . This forms the set of ‘unknown’ statements for the table  $t$ . We do this for all tables in the data.

The above augmentation scheme ensures that the resultant augmented data has an almost equal overall distribution among the three classes. It also ensures that evenness across tables is maintained, as sufficient unknown statements (at least one) are added for each table, and not just overall. After performing the above augmentation scheme, we get 85,296 unknown statements for the autogenerated data and 1,637 unknown statements for the manual data.

### 3.3 Models

#### 3.3.1 Subtask A: The TAPAS Model

TAPAS (Herzig et al., 2020) is a recently released BERT (Devlin et al., 2019) based model that encodes the structural information via column, row and rank embeddings. Eisenschlos et al. extended TAPAS for binary entailment using the TabFact dataset. We use the base variant (12 encoder blocks) of this TAPAS model, which has been pre-trained on the TabFact dataset. We use the base variant over the large variant (24 encoder blocks) due to limitations on computation power and due to the superior performance of the base variant as shown in Section 5.1.2.

For extending the TAPAS model to support the unknown class, we replace the two-neuron linear layer in the pre-trained model by a randomly initialized linear layer consisting of three neurons as the output layer. We then finetune this modified model on the augmented data containing three classes. Figure 2 shows the overview of the modified model. It consists of an embedding layer that concatenates the positional and semantic information of a cell value, the encoder block consisting of transformer layers and subsequent layers that are required for classification output. The core layers of the modified model are in accordance with the original TAPAS model (Herzig et al., 2020; Eisenschlos et al., 2020).

#### 3.3.2 Subtask B: Siamese DistilBERT

For the evidence finding subtask, we finetune the DistilBERT model on the undersampled data. The

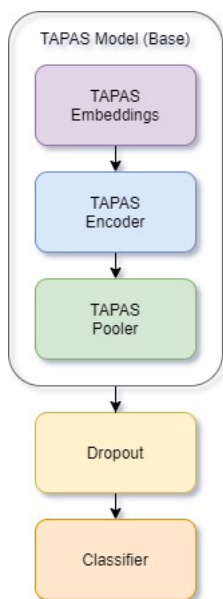


Figure 2: The modified TAPAS architecture. ‘Classifier’ is just a linear layer with three neurons

model was pre-trained with mean pooling on the SNLI dataset, the MultiNLI dataset, and the STS benchmark. The mean pooling is applied in a Siamese setting using the Contrastive loss as the loss function.

Let  $s$  denote a statement, let  $c$  denote the cell, let  $r$  denote the relevancy/label (this is either 0 or 1) and let  $\mathcal{M}(x)$  denote the embedding computed the model for an input  $x$ . Then, the distance  $d$  and the Contrastive loss  $\mathcal{L}(d, r)$  is defined as follows:

$$d = \text{COSINEDISTANCE}(\mathcal{M}(s), \mathcal{M}(c))$$

$$\mathcal{L}(d, r) = \frac{1}{2} \left( (1 - r)d^2 + r(\max(0, m - d))^2 \right) \quad (2)$$

Here,  $m$  denotes the ‘margin’ value, which ensures that the dissimilar pairs contribute to the loss only if their distance is within this margin. For making inferences, we first use our finetuned model to compute the sentence embeddings for the statement and the cell text and then compute the cosine similarity between the two. Hyperparameter tuning and other details about the models are discussed in Section 4.

## 4 Experimental Setup

### 4.1 Hyperparameters and Data Splits

#### 4.1.1 Subtask A

For training, we first merge the autogenerated and manual augmented data. We then perform an 80 – 20 split on this merged data to get our training and

validation sets. We then fine-tune on the training data and validate on the validation set.

The ‘TAPAS Encoder’ (Figure 2) consists of a stack of 12 encoder blocks (similar to the BERT Base). For finetuning, we freeze the entire model, except the last three encoder blocks, the ‘TAPAS Pooler’, and the final classification layer.

This leaves 22M trainable parameters and 89M frozen (i.e. untrainable) parameters.

Categorical Cross-Entropy is used as the loss function. We use a batch size of 32 and finetune for 3 epochs using the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of  $5 \times 10^{-5}$ .

Before making the final submission, the training and development sets are merged together and the model is trained for one more epoch.

#### 4.1.2 Subtask B

We take a 500K sized randomly selected sample from the undersampled data and perform an 80 – 20 split on this to get our training and validation sets. We keep the margin  $m$  as 0.5 and finetune the entire model, with no parameters frozen, for a single epoch using a batch size of 64. For making the inferences from the cosine similarity, we use an empirically selected threshold of 0.3.

## 4.2 Libraries and Tools

Google Colab<sup>2</sup> was used to perform all the experiments. The time taken per epoch for any model did not exceed 10 hours. The GPUs automatically allotted by Colab kept varying between Tesla T4, Tesla P100-PCIE-16GB, and Tesla K80. PyTorch<sup>3</sup> is used as the central framework for both the tasks. For subtask A, Huggingface’s Transformers library<sup>4</sup> was used to load the pre-trained TAPAS model. For subtask B, we use the SentenceTransformers framework<sup>5</sup> to load the pretrained Siamese DistilBERT model.

## 5 Results

### 5.1 Subtask A

#### 5.1.1 Official Metrics

The original pre-trained TAPAS (Base) model, without any finetuning, achieves a two-way F1 score of 62.56. Our final finetuned model achieves

<sup>2</sup>Free version

<sup>3</sup>Version 1.7.0: <https://pytorch.org/docs/1.7.0/>

<sup>4</sup>Version 4.1.1: <https://huggingface.co/transformers/v4.1.1/>

<sup>5</sup>Version 0.4.1: <https://www.sbert.net/>

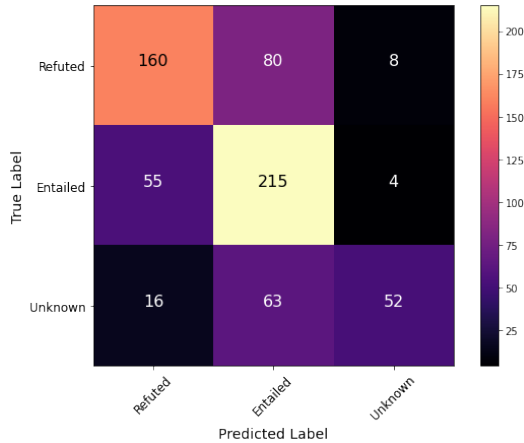


Figure 3: Confusion matrix for the submitted model

a two-way F1 score of 71.72 and a three-way F1 score of 65.59. Figure 3 shows the corresponding confusion matrix. We ranked 8<sup>th</sup> on the official leaderboard for subtask-A.

From the confusion matrix, we observe that the most confusing cases for our model are true ‘unknown’ statements, which get classified as ‘entailed’. On manual analysis of such statements, we observe that many of these have a considerable textual overlap with the table. This misclassification seems to be a consequence of our augmentation scheme. The statements that we add as ‘unknown’ while augmentation have almost no textual overlap with the table data since they were sourced from other tables. Although, in general, as observed in the test data, unknown statements can broadly be classified into two kinds: ‘related’ and ‘unrelated’. The former being unknown statements that are related in a semantic or a directly textual way to the table’s data, and the latter are not related to the table’s data in any way. The following example should make this clear.

Orders	Week 1	Week 2	Week 3
Order 1	Peat	Straw	Silage
Order 2	Straw	Silage	Control
Order 3	Silage	Control	Combo

Table 2: A table containing data about a few weekly orders

In Table 2, a ‘related’ unknown statement will be “A total of 10L of **silage**, **straw**, **peat** or **combo** was distributed”. The boldfaced words overlap directly with the table’s contents

While, an ‘unrelated’ unknown statement can be “Earth revolves around the Sun”. Thus, both of

these sub-classes of the ‘unknown’ class are important. Synthesizing ‘unrelated’ unknown statements is a simple task, whereas synthesizing ‘related’ statements without any additional information seems to be a fairly non-trivial task.

We also observe that wrongly classified statements are, on average, 8.8 characters longer than correctly classified statements. Figure 4 shows this histogram.

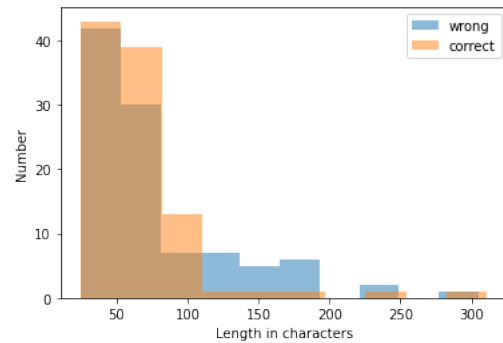


Figure 4: Histogram of lengths (in characters) of 100 randomly selected correctly classified and wrongly classified statements

## 5.1.2 Ablation Study

### Effect of Table Metadata

A table may have additional metadata like the caption, legend, and footer associated with it. For simplicity, we include it directly inside the table as rows. If the table has a caption, we add it as a single row at the top of the table. Similarly, if it has a legend, it goes as a single row below the caption, and the footer is added as a row at the end of the table. Keeping other hyperparameters the same, we get the following results:

Included Metadata	Validation Accuracy
Yes	0.92
No	0.93

Table 3: Impact of inclusion of metadata on validation accuracy

We observe that including the metadata worsens the accuracy. A possible reason could be our way of including the metadata as rows. There may be other better ways of doing this, which may further improve the accuracy.

### Effect of Model Size

We try out two variants of TAPAS: Base and Large. The Base variant has 12 encoder blocks, and the

Large variant has 24, similar to BERT Base and BERT Large. On the TabFact dataset, the Base variant is only about 0.24 points behind the Large one. The following tests are performed on the original, unaugmented data, containing two classes<sup>6</sup>. We merge the original manual and autogenerated statements into a single dataset and perform an 80 – 20 split.

Variant	Validation Accuracy
Base	0.81
Large	0.71

Table 4: Effect of finetuning a larger model

## 5.2 Subtask B

Our final model achieves an F1 score of 43.02. Figure 5 shows the corresponding confusion matrix. As evident from the confusion matrix, there is a lot of room for improvement, especially in handling the ‘relevant’ statements. We ranked 7<sup>th</sup> on the official leaderboard for subtask-B.

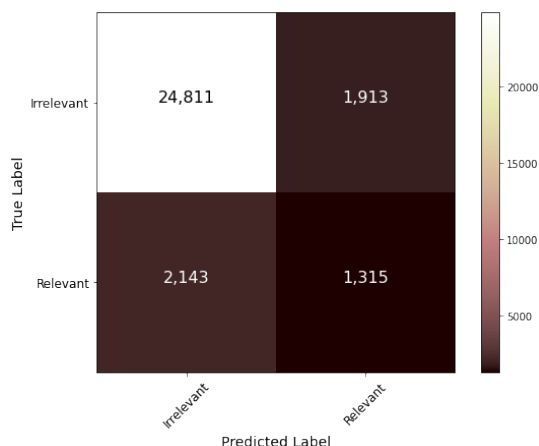


Figure 5: Confusion matrix for Subtask B

## 6 Conclusion

Thus we have presented a statement verification and evidence finding setup for tables. For subtask-A, we extended the TAPAS model to adapt to the ‘unknown’ statements. For subtask-B, we used a semantic approach for evidence finding. Our results for subtask-A show the problems encountered while generating and working with the ‘unknown’ statements. For subtask-B, our results show the effect of taking only the semantic information into

<sup>6</sup>Training the Large variant on augmented data exceeds the time limit of Google Colab (12 hours)

account. An important future prospect for subtask-A would be to find a more effective way of generating the ‘unknown’ statements. For subtask-B, utilizing the table’s available structural information to improve the results seems to be a promising prospect.

## References

- Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. [A large annotated corpus for learning natural language inference](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.
- Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. [SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation](#). In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.
- Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyu Zhou, and William Yang Wang. 2020. [Tabfact: A large-scale dataset for table-based fact verification](#). In *International Conference on Learning Representations*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. [Understanding tables with intermediate pre-training](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics.
- R. Hadsell, S. Chopra, and Y. LeCun. 2006. [Dimensionality reduction by learning an invariant mapping](#). In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, volume 2, pages 1735–1742.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. [TaPas: Weakly supervised table parsing via pre-training](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

- Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. [Search-based neural structured learning for sequential question answering](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics.
- Mayank Jobanputra. 2019. [Unsupervised question answering for fact-checking](#). In *Proceedings of the Second Workshop on Fact Extraction and VERification (FEVER)*, pages 52–56, Hong Kong, China. Association for Computational Linguistics.
- Ilya Loshchilov and Frank Hutter. 2019. [Decoupled weight decay regularization](#). In *International Conference on Learning Representations*.
- Aditya Malte and Pratik Ratadiya. 2019a. Evolution of transfer learning in natural language processing. *arXiv preprint arXiv:1910.07370*.
- Aditya Malte and Pratik Ratadiya. 2019b. Multilingual cyber abuse detection using advanced transformer architecture. In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, pages 784–789. IEEE.
- Panupong Pasupat and Percy Liang. 2015. [Compositional semantic parsing on semi-structured tables](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.
- Pratik Ratadiya and Deepak Mishra. 2019. An attention ensemble based approach for multilabel profanity detection. In *2019 International Conference on Data Mining Workshops (ICDMW)*, pages 544–550. IEEE.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.
- Qi Shi, Yu Zhang, Qingyu Yin, and Ting Liu. 2020. [Learn to combine linguistic and symbolic information for table-based fact verification](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5335–5346, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Nancy Xin Ru Wang, Diwakar Mahajan, Marina Danilevsky, and Sara Rosenthal. 2021. [Semeval-2021 task 9: Fact verification and evidence finding for tabular data in scientific documents \(semtab-facts\)](#). In *Proceedings of the 15th international workshop on semantic evaluation (SemEval-2021)*.
- Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. [A broad-coverage challenge corpus for sentence understanding through inference](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.
- Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. [Program enhanced fact verification with verbalization and graph attention network](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.
- Xinyi Zheng, Doug Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2020. [Global table extractor \(gte\): A framework for joint table identification and cell structure recognition using visual context](#).
- Victor Zhong, Caiming Xiong, and Richard Socher. 2018. [Seq2SQL: Generating structured queries from natural language using reinforcement learning](#).
- Wanjun Zhong, Duyu Tang, Zhangyin Feng, Nan Duan, Ming Zhou, Ming Gong, Linjun Shou, Daxin Jiang, Jiahai Wang, and Jian Yin. 2020. [Logical-FactChecker: Leveraging logical operations for fact checking with graph module network](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6053–6065, Online. Association for Computational Linguistics.

# MedAI at SemEval-2021 Task 10: Negation-aware Pre-training for Source-free Negation Detection Domain Adaptation

Jinquan Sun, Qi Zhang, Yu Wang, Lei Zhang

Alibaba Group Inc.

{jinquan.sjq, mickey.zq, tonggou.wangyu}@alibaba-inc.com  
lei.zhang.lz@alibaba-inc.com

## Abstract

Due to the increasing concerns for data privacy, source-free unsupervised domain adaptation attracts more and more research attention, where only a trained source model is assumed to be available, while the labeled source data remains private. To get promising adaptation results, we need to find effective ways to transfer knowledge learned in source domain and leverage useful domain specific information from target domain at the same time. This paper describes our winning contribution to SemEval 2021 Task 10: *Source-Free Domain Adaptation for Semantic Processing*. Our key idea is to leverage the model trained on source domain data to generate pseudo labels for target domain samples. Besides, we propose Negation-aware Pre-training (NAP) to incorporate negation knowledge into model. Our method wins the 1st place with F1-score of 0.822 on the official blind test set of *Negation Detection Track*.

## 1 Introduction

The *Negation Detection Track* of SemEval 2021 Task 10: *Source-Free Domain Adaptation for Semantic Processing* provides a new setting for unsupervised domain adaptation task which ask participants to conduct negation detection in target domain only with model trained on source domain (namely source model) and unlabeled target domain data. Negation detection is a span-in-context classification problem, where the model will jointly consider both the target mention to be classified and its surrounding context. For example, in sentence *Has no <e> diarrhea <e\> and no new lumps or masses*, the target span *diarrhea* is negated by its surrounding context *no*. This task is important for physicians to extract key information from clinical text. The test dataset used is based on MIMIC-III version 1.4 (Johnson et al., 2016), which is a large, freely-available english database comprising de-identified health-related data.

We approach this task as a problem of learning with pseudo labels. Our main interests include 1) negation knowledge infusion through pre-training on target domain and 2) high-quality pseudo label generation. We divide the task into two stages: Negation-aware Pre-training (NAP) stage and Pseudo label Training stage. In the NAP stage, token-level and sentence-level negation semantic are embedded into model. In the pseudo label training stage, confidence threshold search and mean self-entropy are used to select target domain samples with highly confident pseudo labels.

## 2 Related Work

Traditional negation detection method are mostly rule-based. These methods (Chapman et al., 2001; Sanchez-Graillet and Poesio, 2007; Huang and Lowe, 2007; Sohn et al., 2012) used regular expression algorithm, dependency parsing and grammatical parsing to perform negation cue detection and scope resolution. Recent years, deep learning has been applied to negation detection task. In (Qian et al., 2016), Convolutional Neural Network was used to recognize negation scope in the sentence. (Lazib et al., 2019) and (Gautam et al., 2018) leveraged recurrent neural network variants to perform negation scope resolution and achieved better performance with BiLSTM, which further indicates the potential in deep learning-based methods. Joint model to detect negation cues and targets simultaneously had been studied by Bhatia et al. (2019). More recently, popular transformer-based model (Khandelwal and Sawant, 2019) had also been used to perform negation detection.

Due to data privacy and data transmission problem, several source-free unsupervised domain adaptation methods (Liang et al., 2020; Kim et al., 2020; Yang et al., 2020) have been proposed for image classification task. These methods mostly focus

on generating high-quality pseudo labels for target domain samples before or during training phase and do not involve self-supervised pre-training.

In the natural language processing field, pre-training is popular. We train language models on huge corpora and fine-tune the pre-trained architectures (Devlin et al., 2018; Liu et al., 2019; Zhang et al., 2019; Yang et al., 2019) in downstream tasks, achieving state-of-the-art results on most NLP tasks. Prior studies (Radford et al., 2018; Chronopoulou et al., 2019; Gururangan et al., 2020; Lee et al., 2020) has further shown the potential of domain-adaptive and task-adaptive pre-training.

### 3 Method

We approach the task of source-free domain adaptation for negation detection as a problem of learning with pseudo labels. To generate high-quality pseudo labels, we use mean self-entropy as metric to search appropriate probability threshold, which is inspired by (Li et al., 2020). Besides, in order to learn more negation semantic knowledge from target domain, we propose negation-aware pre-training to incorporate negation knowledge by self-supervised training.

#### 3.1 Negation-aware Pre-training

In prior studies, negation cues are important for rule-based and machine learning-based methods. We propose Negation-Aware Pre-training NAP to embed the knowledge of negation cues into representation. As shown in Figure 1, common token masking, negation cue prediction and pseudo negation detection are included in NAP. Common token masking is conducted to capture target domain language knowledge. Negation cue prediction could help the model recognize token-level negation information based on collected negation cue lexicon. Pseudo negation detection is a sequence classification task and designed for complex sentence-level negation knowledge. It is the same as our final negation detection task, however, the target mention and corresponding label is generated by simple heuristic rules. Although these data is somewhat noisy, with the help of pseudo negation detection pre-training, more negation information could be embedded into model.

**Negation Cue Lexicon** Negation cues are key to the Negation-Aware Pre-training. Based on the lexicon created by (Weng et al., 2020), we divide negation cues into 2 categories: Pre-negation

PREN	POSN
not	unlikely
none	be ruled out
nor	be excluded
without	be resolved
deny	be absent
no evidence of	be negative

Table 1: Examples of negation cues.

(PREN), Post-negation (POSN). Pre-negation and Post-negation mean the negation cues locate before or after the target mentions respectively. Table 1 shows several examples of each type of negation cues.

**Pseudo Pre-training Data Generation** To perform pseudo negation detection task, we need a large number of labeled data. We design simple yet effective rules to generate training data. To simplify the process, we assume all clinical event mentions are single noun. For sentence *without evidence of diarrhea, vomit*, we first locate the negation cue *without*. Since *without* is a pre-negation cue, we take the 3 tokens behind it ( *evidence of diarrhea* ) as target context. In the target context, the furthest noun *diarrhea* from *without* is selected as target mention. Finally, we generate a pseudo training data: *without evidence of <e> diarrhea <e\>, vomit* with negated label. For samples with post-negation cue, the process is similar, but the target context is before the negation cue. Generally, the selected target mentions are negated by surrounding cues, but there is a special case: double negative. For instance, in sentence *The report can not rule out diarrhea*, *diarrhea* is not negated, since *rule out* is negated by *not*. We assign non-negated label for these double-negative cases. We also generate more non-negated samples by randomly select sentences including no negation cues.

**Objectives of Pre-training** As illustrated in Figure 1, there are 3 tasks included in the pre-training stage. Common token masking is inherited from RoBERTa (Liu et al., 2019). It is used to capture low level language information in test domain. Specifically, during pre-training, except for the tokens in negation cue lexicon, about 15% of input tokens are random sampled and masked. The model is trained to recover original tokens in the corrupted input sequence. We denote the objective of common token masking task as  $L_{mlm}$ . Negation cue prediction as a token classification task is import



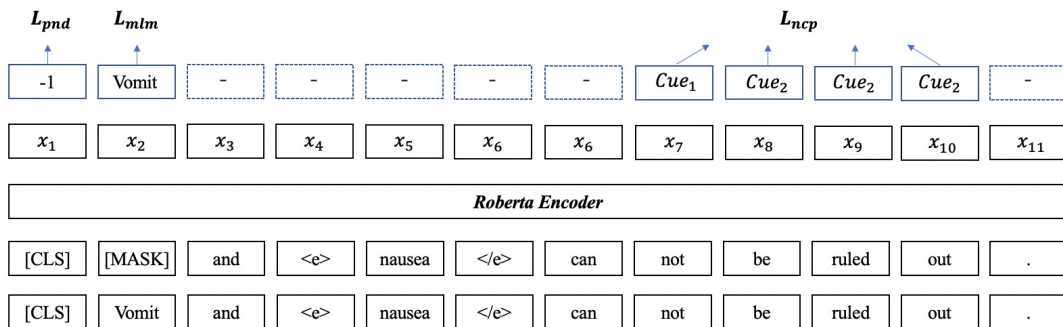


Figure 1: The framework of NAP, Negation-Aware Pre-training. The NAP includes 3 pre-training tasks: common token masking, negation cue detection and pseudo negation detection. Common token masking is inherited from prior pre-training work. Negation cue detection is a token classification task, which aims to embed negation knowledge at token level. Pseudo negation detection is similar to final negation detection task, but the training data is generated based on heuristic rule. Pseudo negation detection could help model capture the complex relation between negation cue and target mention.

for negation knowledge infusion. For each token in input sequence, the task aims to guide the model predict its negation polarity (negation or not) based on its representation  $x_i$  output by transformer encoder. Through this way, the pre-trained model could learn the negation knowledge at token level. The objective of negation cue prediction is the same as classical token classification task and is denoted as  $L_{ncp}$ . Pseudo negation detection is another crucial key to negation-aware pre-training. Compared to negation cue prediction task, this task could further help model to capture the semantic relation between negation cues and target mentions. The objective  $L_{pnd}$  for a single sample is defined as follows:

$$\hat{y} = \text{sigmoid}(Wx_1 + b)$$

$$L_{pnd} = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

Here  $x_1$  denotes the output vector of first token from transformer encoder. All the above 3 pre-training objectives are jointly optimized. Thus, overall pre-training objective  $L$  is:

$$L = L_{mlm} + L_{ncp} + L_{pnd}$$

### 3.2 Pseudo label Training

Although the negation knowledge is infused through pre-training tasks, the pre-trained model is still lack of the ability to understand complicated semantic information and negation relation. Since the training data and corresponding labels in pre-training stage are generated by simple heuristic rules, most training sample are easy to learn and some samples with wrong labels may harm the model. Thus, test domain samples with high-

quality labels are needed to guide the pre-trained model learn more useful information. We leverage the source model to predict the probability of each test sample to be negated or non-negated. Then inspired by *Self-entropy Descent* (SED) proposed in (Li et al., 2020), we conduct *Confidence Threshold Search* to generate high-quality test domain samples.

**Confidence Threshold Search** Self-entropy could be used as a metric to measure the prediction uncertainty (Kim et al., 2020), i.e.  $H(x) = -\sum p(x) \log(p(x))$ . The lower the self-entropy the more confident the prediction is. We set the probability threshold to be non-negated as 0.999 empirically, then search the probability to be negated from 0.985 to 0.975. The step size is set to be 0.001. The generated pseudo labels are used to fine-tune the source model and then evaluate the mean self-entropy of the dataset after training. When the mean self-entropy descends and hits the first local minimum, we take the corresponding probability as an appropriate threshold for generating labels. Samples do not reach the threshold will be excluded.

## 4 Source Model and Data

**Source Domain Data** The source domain data is from SHARP Seed dataset which consists of de-identified clinical notes from Mayo Clinic. In the SHARP data, clinical events are marked with a boolean polarity indicator, with values of either asserted or negated. There are 10259 samples provided including 902 negated instances for source model training.

**Source Model** Since the data privacy limitation, SHARP Seed dataset cannot be distributed. Thus organizers provide a "span-in-context" negation detection model trained on SHARP Seed dataset as the source model. The source model is RoBERTa-based and could achieve promising result on the SHARP Seed dataset.

**Target Domain Data** The target domain data is from MIMIC-III version 1.4 dataset (Johnson et al., 2016). MIMIC-III version 1.4 is a large, freely-available database comprising de-identified health-related data associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. Based on rules, we extract about 50,000 pseudo samples from the file *NOTEEVENTS.csv* to perform negation-aware pre-training. The official test dataset including 9580 samples is also extracted from the file *NOTEEVENTS.csv*. To further validate the effectiveness of the proposed method, we further create a custom test dataset which includes 500 negated samples and 500 non-negated samples. In the custom test dataset, various negation cues and negation style are included.

## 5 Experiment

### 5.1 Negation-aware Pre-training Stage

We leverage the source model provided by organizers as our initial model in pre-training stage. Besides, several layers are added to perform masked language modeling and negation cue prediction. During pre-training, the learning rate is set as 0.00001, and batch size is set to be 64. We conduct pre-training in 3 epochs.

### 5.2 Pseudo Label Training Stage

We assume that a sample including no negation cue is definitely non-negated. Thus, we assign these sample without negation cues non-negated label. These samples will not be included in the training phase. Since the provided test samples are extracted directly from *NOTEEVENTS.csv* file, the format of each sample is messy, we conduct sentence split with NLTK toolkit (Loper and Bird, 2002) for each sample and only keep the sentence with target mention.

Then confidence threshold search is conducted to generate high-quality labels for the rest of test data. Finally, the confidence threshold for negated and non-negated samples are selected as 0.983 and 0.999 respectively. In other words, if a test sam-

ple assigned negated label, the probability to be negated generated by the source model should be higher than 0.983. Similarly, if a test sample assigned non-negated label, the probability to be non-negated should be higher than 0.999.

During training stage, we use the source model provided by organizers but initialize the transformer encoder with the corresponding part from pre-training model, because the transformer encoder from the pre-training model could capture various negation knowledge from input sentences. The learning rate is kept as 0.00001, and batch size is 32. The number of epoch is set to be 5. In the first 2 epoch, the parameters of *ClassificationHead* in model is frozen.

## 6 Results

The result is evaluated using the standard precision, recall and F1 scores as used in most published work. We achieve the best performance on the official blind test dataset.

### 6.1 Result on Official Test Dataset

We compare our method with two baselines, and the result is shown in Table 2. The result of source model without any domain adaptation is inferior. Masked language modeling trained on target domain data could improve the performance. However, its effect is not significant, because masked language pre-training focus more on low-level language information rather than high-level semantic knowledge about negation. With the help of NAP, the adapted model could improve the recall score with a large margin. This indicates that our negation-aware pre-training method could help embed negation knowledge into the sequence representation, and facilitate the domain adaptation from source domain to target domain.

Although the proposed adaptation method achieves superior result in the competition, there still exists a problem which harms the recall performance: the adapted model is not sensitive to long-term negation dependency. For example, in the case *He denies chest pain, dyspnea, dizziness/lightheadedness, or <e> abdominal pain <e\>*, though the mention *abdominal pain* is connected with negation cue *denies* via *or*, the model still output non-negated prediction. This phenomenon may be caused by pre-training, since in pseudo negation detection pre-training task, the training data are generated only based on simple

Methods	F1	Precision	Recall
Source	0.685	0.921	0.545
Source + MLM	0.724	0.905	0.603
Source + NAP	<b>0.822</b>	<b>0.902</b>	<b>0.756</b>

Table 2: Results of different methods on official test set. MLM, NAP denote masked language modeling (common token masking), negation-aware pre-training respectively. Our experiments are conducted on the data cleaned, so the result of source model without adaptation is better than the one organizers provide.

Methods	Precision	Recall
Source	0.761	0.502
Source + MLM	0.783	0.526
Source + NAP	0.894	0.833

Table 3: Results of different methods on customized test set.

rules: the clinical mention is a single noun and the distance between mention and negation cue is limited to no more than 3 tokens. To handle this problem, useful and effective generation method should be further explored. In addition, dependency relation between tokens may be introduced into both pre-training and training stage to solve this problem in negation detection task.

## 6.2 Result on customized Test Dataset

Negated samples in the official test dataset only include *deny*, *none*, *no*, *not* and *without*, so we also conduct experiments on the customized test data we manually created from test domain, which contains various negation cues and two negation styles (active or passive voice). As shown in Table 3, compared to the proposed method, the recall of source model and source model with MLM is much lower, because many negated samples with *never*, *resolve*, *free of*, *absent* and *exclude* can not be recognized correctly. Meanwhile, due to the lack of the ability to capture double-negative semantic, they both fail to distinguish false-positive samples from real positive ones. However, with the negation knowledge embedded through negation-aware pre-training, our method could handle both scenarios better.

## 7 Conclusion

In this paper, we model source-free domain adaptation as learning with pseudo label. We leverage mean self-entropy of dataset to search appropriate probability threshold for high-quality pseudo label generation. Besides, we propose negation-aware

pre-training to integrate different types of negation knowledge to improve the generalization of negation representation. The result shows that the additional negation-aware pre-training is helpful for negation detection task. In the future, we will work towards more robust pseudo label generation method and effective pre-training task introducing more knowledge such as part-of-speech tag and dependency relation.

## References

- Parminder Bhatia, E Busra Celikkaya, and Mohammed Khalilia. 2019. End-to-end joint entity extraction and negation detection for clinical text. In *International Workshop on Health Intelligence*, pages 139–148. Springer.
- Wendy W Chapman, Will Bridewell, Paul Hanbury, Gregory F Cooper, and Bruce G Buchanan. 2001. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of biomedical informatics*, 34(5):301–310.
- Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. 2019. An embarrassingly simple approach for transfer learning from pretrained language models. *arXiv preprint arXiv:1902.10547*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Dipesh Gautam, Nabin Maharjan, Rajendra Banjade, Lasang Jimba Tamang, and Vasile Rus. 2018. Long short term memory based models for negation handling in tutorial dialogues. In *FLAIRS Conference*, pages 14–19.
- Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.
- Yang Huang and Henry J Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. *Journal of the American medical informatics association*, 14(3):304–311.

- Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9.
- Aditya Khandelwal and Suraj Sawant. 2019. Negbert: A transfer learning approach for negation detection and scope resolution. *arXiv preprint arXiv:1911.04211*.
- Youngeun Kim, Sungeun Hong, Donghyeon Cho, Hyoungeob Park, and Priyadarshini Panda. 2020. Domain adaptation without source data. *arXiv preprint arXiv:2007.01524*.
- Lydia Lazib, Yanyan Zhao, Bing Qin, and Ting Liu. 2019. Negation scope detection with recurrent neural networks models in review texts. *International Journal of High Performance Computing and Networking*, 13(2):211–221.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.
- Xianfeng Li, Weijie Chen, Di Xie, Shicai Yang, Peng Yuan, Shiliang Pu, and Yueting Zhuang. 2020. A free lunch for unsupervised domain adaptive object detection without source data. *arXiv preprint arXiv:2012.05400*.
- Jian Liang, Dapeng Hu, and Jiashi Feng. 2020. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 6028–6039. PMLR.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.
- Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.
- Zhong Qian, Peifeng Li, Qiaoming Zhu, Guodong Zhou, Zhunchen Luo, and Wei Luo. 2016. Speculation and negation scope detection via convolutional neural networks. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 815–825.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.
- Olivia Sanchez-Graillet and Massimo Poesio. 2007. Negation of protein–protein interactions: analysis and extraction. *Bioinformatics*, 23(13):i424–i432.
- Sunghwan Sohn, Stephen Wu, and Christopher G Chute. 2012. Dependency parser-based negation detection in clinical narratives. *AMIA Summits on Translational Science Proceedings*, 2012:1.
- Wei-Hung Weng, Yu-An Chung, and Schrasing Tong. 2020. Clinical text summarization with syntax-based negation and semantic concept identification. *arXiv preprint arXiv:2003.00353*.
- Shiqi Yang, Yaxing Wang, Joost van de Weijer, and Luis Herranz. 2020. Unsupervised domain adaptation without source data by casting a bait. *arXiv preprint arXiv:2010.12427*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.

# YNU-HPCC at SemEval-2021 Task 10: Using a Transformer-based Source-Free Domain Adaptation Model for Semantic Processing

Zhewen Yu, Jin Wang and Xuejie Zhang

School of Information Science and Engineering

Yunnan University

Kunming, China

zhwyu@mail.ynu.edu.cn, {wangjin, xjzhang}@ynu.edu.cn

## Abstract

Data sharing restrictions are common in natural language processing datasets. The aim of this study is to develop a model that is trained in a source domain to make predictions for a target domain with respect to domain data. To address this problem, the organizers provided models that fine-tuned a large number of source domain data on pre-trained models and dev data for participants. However, source domain data were not distributed. This paper describes the model provided for the name entity recognition task and ways to develop the model. Because little data are provided, pre-trained models are suitable for solving cross-domain tasks. The models fine-tuned by a large number of other domains could be effective in the new domain because the task did not change. The code of this paper is available at <https://github.com/windforfuture/SemEval-2021-Task10>.

## 1 Introduction

Data sharing constraints are common in natural language processing (NLP) datasets. For example, Twitter policies prohibit the sharing of tweet text, although tweet IDs may be shared. In clinical NLP, the situation is even more prevalent because information on patients' health must be protected. Obtaining annotations about health texts often requires the signing of complex data usage agreements. During the competition, the organizers provided models which fine-tuned on the annotated source domain data, while the source domain data could not be distributed. The organizers also provided labeled data as dev data and unlabeled target domain data as test data.

There were two sub-tasks for SemEval Task 10: 1) to classify clinical event mentions (e.g., diseases, symptoms, procedures, etc.) for whether

they are being negated by their context. This is a text-classification task (Yuan et al., 2020). 2) to find time expressions (Laparra et al., 2018) in text, this is a sequence-tagging task. SemEval Task 10 was different from traditional NLP tasks, where training and testing were in the same domain. Predictions could be out of control due to the different target domain. The domain of the dev data is related to the source and target domains, therefore the model can be developed. In SemEval Task 10, it is necessary to develop an existing model along with training the model with labeled data or unlabeled data. Furthermore, the model can be developed by fine-tuning it in different ways.

For certain reasons, only the results for the second subtask were submitted, that is, time expression recognition, which can be considered as a name entity recognition (NER) task. The participants were asked to find time expressions in the text through the task. This is a sequence-tagging task that uses fine-grained time expression annotations that are a component of SemEval 2018 Task 6 (Laparra et al., 2018). To deal with this task, deep learning models, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997), bidirectional LSTM with conditional random field (BiLSTM-CRF) (Huang et al., 2015), and bidirectional encoder representation from transformers (BERT) (Dai et al., 2019) have been developed to challenge the NER task. Owing to the lack of a training set, pre-trained models (Qiu et al., 2020) were considered. The NER task can become difficult as the number of classifications increase. In this task, 33 types of entities need to be recognized, and phrases and words may be one of the types. For the base model, B-prefix and I-prefix were used to discriminate the beginning of classification and ensure the accuracy of the tokens; therefore, 65 types of entities were used in the model. In this paper, an ensemble model is proposed for time

expression recognition with a hard voting strategy. Each input sample was first tokenized using the matched model tokenizer. Base models, such as BERT (Devlin et al., 2019) and its variants, including RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019) and ALBERT (Lan et al., 2019), were used to learn hidden representations for each token. Then, a fully connected dense layer with a *Softmax* function was used for classification. By using a hard voting strategy, the predictions from different base models were merged with the final result. Experimental results show that the proposed ensemble models achieve a competitive result with the official baseline model, which ranked fifth in sub-task 2.

The remainder of this paper is organized as follows. Section 2 describes the overall structure of the proposed ensemble model. The comparative experimental results and discussion are presented in Section 3. Finally, conclusions are presented in Section 4.

## 2 Ensemble Model for Source-Free Domain Adaptation

The official baseline model is a fine-tuned RoBERTa model: clulab/roberta-timex-semeval. It uses RoBERTa (Liu et al., 2019) for token classification architecture, which is pre-trained using no next-sentence prediction (NSP) and dynamic masking. The RoBERTa model achieved better performance than the BERT model because it was pre-trained by larger volume data, more steps, larger batches, and larger vocabulary than the BERT model. In addition, the provided baseline model was fine-tuned with approximately 25,000 expressions in de-identified clinical notes as well as the development dataset (Sanh et al., 2019). Four other models were also implemented using a hard voting strategy and a hard voting result for a single submission.

### 2.1 Tokenization

Transforming words to vectors is a necessary step in NLP tasks. Different word representations were used in our implementation, including word2vec (Mikolov et al., 2013; CHURCH, 2017), GloVe (Pennington et al., 2014), ELMo (Peters et al., 2018), and BERT (Devlin et al., 2019). For the RoBERTa model, we used only the matched RoBERTa tokenizer to build word vectors with a length of 514. Given a sentence  $\mathbf{x} =$

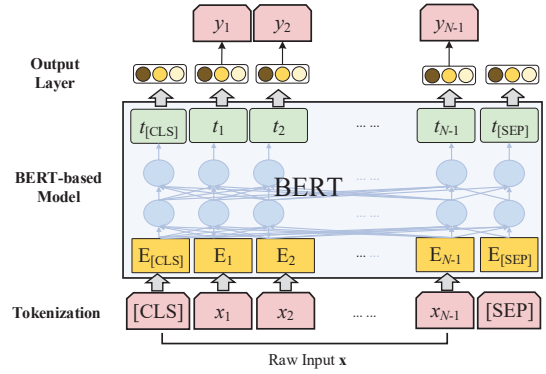


Figure 1: Overall architecture of the base model.

$[x_1, x_2, \dots, x_M]$  of length  $M$ , different lengths of the sentence will result in different lengths of the representation. Therefore, we considered the maximum sentence length as  $N - 1$ . If the length was less than  $N - 1$ , then it was padded with zero values to make it equal to  $N - 1$ . For each input, two specific tokens were added to the beginning and end of the raw input, that is,  $\langle s \rangle$  and  $\langle /s \rangle$  (or [CLS] and [SEP]). The RoBERTa tokenizer uses byte-pair encoding to obtain more relations and meanings. Then, these are converted into a sequence of subwords, which are then mapped into token, position, and segment embeddings, that is,  $[E_{[CLS]}, E_1, \dots, E_{N-1}, E_{[SEP]}]$ . In the proposed model shown in Fig.1, the raw inputs were split into one or more 514-dimensional vectors according to the number of sentences contained in the corresponding input.

### 2.2 BERT-based Model

To extract the semantic features, a pre-trained language model (Qiu et al., 2020) was used. It achieved impressive performance in various NLP tasks. It contains multiple layers of bidirectional transformer encoders (Vaswani et al., 2017) and is then pre-trained by using unsupervised learning of either the masked language model (with a masked ratio of 15%) or the NSP.

The aforementioned pre-trained language model contains 12 layers of transformers with a hidden size of 768. Then, the embeddings of both contexts are fed into a BERT model to obtain the semantic representation  $T \in \mathbb{R}^{dt}$ , denoted as

$$\begin{aligned} T^F &= [t_{[CLS]}, t_1, \dots, t_{N-1}, t_{[SEP]}] \\ &= f_{BERT}([E_{[CLS]}, E_1, \dots, \\ &\quad E_{N-1}, E_{[SEP]}]; \theta_{BERT}) \quad (1) \end{aligned}$$

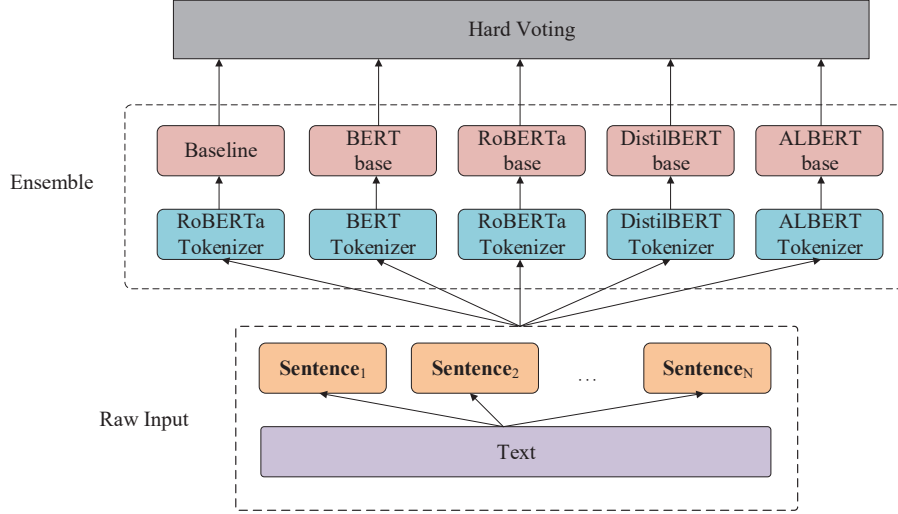


Figure 2: The ensemble of models.

where  $\theta_{BERT}$  is the trainable parameter of the BERT model, which is then fine-tuned during model training, and  $d_t = 768$  is the dimensionality of the hidden representation.

### 2.3 Output Layer

The token classification in the BERT model architecture is considered as a series of binary classifications. For each token, the classification was a one-layer MLP with a *Softmax* function. The loss function is a categorical cross-entropy, defined as:

$$\hat{y}_m^c = \text{softmax}(W_t t_m + b_t) \quad (2)$$

$$L = - \sum_{k=1}^K \sum_{c=1}^C \sum_{m=1}^M y_m \log(\hat{y}_m^c) \quad (3)$$

where  $y_m$  and  $\hat{y}_m^c$  denote the gold label and the predicted probability of samples  $k$ , respectively,  $K$  and  $C$  are the number of training samples and candidate categories, respectively;  $W_t$  and  $b_t$  are the weight and bias, respectively, which are associated with the fully connected layer. The entire network was trained by back-propagation (Rumelhart et al., 1986) while the BERT model was fine-tuned with the provided labeled data in the training phase.

### 2.4 Ensemble Learning

To output the final result, we used a hard voting strategy to integrate the results from different base models, including the official baseline, BERT, RoBERTa, DistilBERT, and ALBERT models, as shown in Fig. 2. In hard voting, every individual classifier votes for a class and the majority wins.

In statistical terms, the predicted target label of the ensemble is the mode of the distribution of the individually predicted labels.

## 3 Experimental Results

### 3.1 Datasets

The organization provided 99 news articles and matched 99 annotated data files for subtask 2. There were different types in these articles, such as ABC, APW, and CNN. The dev data provided were related to news, and the source data were related to medical data. The test data were related to the food security. The cross-domain task consisted of three domains. Each raw input was divided into sentences. The annotated data files were formed as XML files. Every XML file contained a few entities that consisted of ids, spans, and types. Span contains the start position, and the end positions and types are classified. The labels should be transformed because of the offset that were used as the input data.

### 3.2 Evaluation Metrics

Subtask 2 was evaluated using standard precision, recall, and mainly the  $F_1$ -score. The  $F_1$ -score is often used to evaluate unbalanced data, and is defined as follows:

$$F_1\text{-score} = 2 * \frac{P * R}{(P + R)} \quad (4)$$

where  $P$  and  $R$  denote the precision and recall, respectively. A higher  $F_1$ -score indicates better model prediction performance.

Model	Vocab Size	Position Embeddings	Attention Heads	Hidden Layers
clulab/roberta-timex-semeval	50265	514	12	12
BERT base	30522	512	12	12
RoBERTa base	50265	514	12	12
DistilBERT base	30522	512	12	6
ALBERT base	30000	512	12	12

Table 1: Optimal parameter settings of the base models

Model	R1	R2	R3	R4	R5	Avg.
clulab/roberta-timex-semeval	<b>0.864</b>	<b>0.910</b>	0.883	0.857	0.859	<b>0.875</b>
BERT base	0.844	0.880	0.895	0.847	0.837	0.861
RoBERTa base	0.817	0.850	0.843	0.809	0.823	0.828
DistilBERT base	0.828	0.868	0.874	0.837	0.839	0.849
ALBERT base	0.809	0.811	0.752	0.792	0.774	0.788
Hard Voting	0.856	0.889	<b>0.900</b>	<b>0.866</b>	<b>0.860</b>	0.874

Table 2: Empirical results of the ensemble model and the base models

### 3.3 Implementation Details

We fine-tuned the official baseline model (RoBERTa) and four other models: the BERT model (bert-base-uncased), RoBERTa model (roberta-base), DistilBERT model (distilbert-base-uncased), and ALBERT model (albert-base-v2). For each model, 5-fold cross-validation was performed. For each run, the datasets were split into a ratio of 8:2 for the training and dev sets.

We trained the new model by adding the same label2id and id2label JSON as the provided model to the original “config.json” We ran codes using the downloaded model and the modified “config.json.” After training, the models were used for prediction, and the results were recorded. After comparing the results, we observed that the model provided the best performance. Then, we performed hard voting that picked out one result, if three or more results of these models were the same.

To obtain additional results, we split the provided data to obtain different dev data and test data five times, and the test data were different every time. In every round, the initial models were used, and five rounds were tested. Hard voting gave the best results, followed by the provided model. Finally, we fine-tuned these initial models with the test data in the evaluation phase and chose the results of the provided model and the hard voting for submissions.

### 3.4 Fine-tuning the Parameters

To train the proposed model, the Adam optimizer applied a warmup strategy with a weight decay of 0.01 during training. The learning rates of all the base models were 5e-6. For the official baseline model, the default parameters were used in our experiments because the initial parameters usually performed well in other experiments. For other models, we fine-tuned the hyperparameters using a grid-search strategy. Once the optimal settings of the parameters were obtained, they were used for classification on the test sets. The details of the hyperparameters are summarized in Table 1.

### 3.5 Comparative Results

Table 2 shows the results of the base models on different folds, that is, R1–R5. The results showed that the hard-voting ensemble model outperformed the official baseline model with three rounds, but the average  $F_1$ -score of the ensemble model was less than the average  $F_1$ -score of the official baseline model. Considering that the evaluation data is in a new domain, we finally submitted the result of the official baseline model, which was fine-tuned by feeding numerous annotated data in the source domain and the result of the hard voting ensemble model. For the test set, the newly released official baseline model, that is, **Organizers (new)**, outperformed the previously released baseline model, that is, **Organizers (previous)**, which was only pre-trained on the source data and achieved an  $F_1$ -score of 0.794 (see Table 3). Owing to the low amount of



Team Name	$F_1$ -score
BLCUFIGHT-1	0.815
Self-Adapter-1	0.811
BLCUFIGHT-2	0.810
Baseline-2	0.804
<b>YNU-HPCC-2</b>	<b>0.803</b>
Self-Adapter-2	0.797
PTST-UoM-1	0.796
UArizona-1	0.795
UArizona-2	0.795
Boom-1	0.795
Baseline-1	0.794
KISNLP-1	0.793
KISNLP-2	0.781
<b>YNU-HPCC-1</b>	<b>0.748</b>

Table 3: All results on leaderboard for time expression recognition.

data used for fine-tuning, the performance of the proposed model is slightly lower than that of the new official baseline model. However, its performance is still competitive and finally ranked fifth on the leaderboard.

## 4 Conclusions

The SemEval-2021 Task 10 framework requests participants to develop semantic annotation systems in the face of data sharing constraints. In this study, we fine-tuned the official baseline model and then combined it with four other pre-trained models with a hard voting strategy for time expression recognition. Experimental results showed that the proposed model outperformed the previously released baseline model, achieved a competitive result with the newly released official baseline model, and finally ranked fifth on the leaderboard. Future work will attempt to improve the performance of cross-domain NER tasks.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants Nos. 61702443, 61966038 and 61762091.

## References

KENNETH WARD CHURCH. 2017. *Word2Vec*. *Nat. Lang. Eng.*, 23(1).

Zhenjin Dai, Xutao Wang, Pin Ni, Yuming Li, Gangmin Li, and Xuming Bai. 2019. Named Entity

Recognition Using BERT BiLSTM CRF for Chinese Electronic Health Records. *Proc. - 2019 12th Int. Congr. Image Signal Process. Biomed. Eng. Informatics, CISP-BMEI 2019*, (October 2020).

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, 1(Mlm):4171–4186.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8).

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv*, pages 1–17.

Egoitz Laparra, Dongfang Xu, Ahmed Elsayed, Steven Bethard, and Martha Palmer. 2018. *SemEval 2018 Task 6: Parsing Time Normalizations*. pages 88–96.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, (1).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.*, pages 1–9.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *NAACL HLT 2018 - 2018 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Conf.*, 1:2227–2237.

Xi Peng Qiu, Tian Xiang Sun, Yi Ge Xu, Yun Fan Shao, Ning Dai, and Xuan Jing Huang. 2020. Pre-trained models for natural language processing: A survey. *Sci. China Technol. Sci.*, 63(10):1872–1897.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088).

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv*, pages 2–6.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Adv. Neural Inf. Process. Syst.*, 2017-Decem(Nips):5999–6009.

Li Yuan, Jin Wang, Liang-Chih Yu, and Xuejie Zhang. 2020. Graph Attention Network with Memory Fusion for Aspect-level Sentiment Analysis. *Proc. 1st Conf. Asia-Pacific Chapter Assoc. Comput. Linguist. 10th Int. Jt. Conf. Nat. Lang. Process.*, pages 27–36.

# ECNUICA at SemEval-2021 Task 11: Rule based Information Extraction Pipeline

Jiaju Lin, Jing Ling\*, Zhiwei Wang\*, Jiawei Liu\*, Qin Chen, Liang He

School of Computer Science and Technology

East China Normal University

51205901095@stu.ecnu.edu.cn

## Abstract

This paper presents our endeavor for solving task11, NLPContributionGraph, of SemEval-2021. The purpose of the task is to extract triples from a paper in the Nature Language Processing field for constructing an Open Research Knowledge Graph. The task includes three sub-tasks: detecting the contribution sentences in papers, identifying scientific terms and predicate phrases from the contribution sentences; and inferring triples in the form of (subject, predicate, object) as statements for Knowledge Graph building. In this paper, we apply an ensemble of various fine-tuned pre-trained language models (PLM) for tasks one and two. In addition, the self-training methods are adopted for tackling the shortage of annotated data. For the third task, rather than using classic neural open information extraction (OIE) architectures, we generate potential triples via manually designed rules and develop a binary classifier to differentiate positive ones from others. The quantitative results show that we obtain the 4<sup>th</sup>, 2<sup>nd</sup>, and 2<sup>nd</sup> rank in three evaluation phases.

## 1 Introduction

The notion of Open Research Knowledge Graph (ORKG) is first proposed by (Jaradeh et al., 2019) who take steps toward a knowledge graph based infrastructure that acquires scholarly knowledge in machine actionable form. In that form, researchers can keep up with cutting edge academic achievements and eliminate cognitive overload. To accelerate the construction of ORKG, an automatic system is expected. The SemEval-21 task11 is a triple extraction task targeted at building that system. As shown in Table1, the task is divided into three sub-parts corresponding to different processing steps: Sub-task A detects contribution sen-

tences in English articles and classifies them into information units such as Approaches, Models, and Ablation – analysis; Sub-task B extracts scientific terms and relational cue phrases from contribution sentences; Sub-task C infers subject-predicate-object triples for KG building with the results of two previous sub-tasks.

For Sub-task A—contribution sentence detection—the evaluation data covers a larger sphere than the training data. Additionally, the amount of annotated samples differs among research fields. Hence, we use self-training to generate a set of silver samples for fields lacking gold data. An ensemble of fine-tuned PLM based classifiers is then deployed to categorize sentences. For sub-task B—scientific term extraction—we compared BERT based sequence labeling systems in detail and chose the best architecture. For sub-task C—triple generation—we give insight into the construction of triples and designed a rule for potential triples generation. A binary classifier is then applied to distinguish the positive triples.

Our quantitative results show data augmentation via self-training is of paramount importance for sub-task A. Although seldom is CRF used with transformer-based language models together, in the system for sub-task B, an additional CRF layer after a RoBERTa based encoder can still boost performance. In sub-task C, popular neural information extraction models are inferior to the rule based methods.

## 2 Background

### 2.1 Data description

The training process is developed on the dataset provided by the SemEval21 Task11. The training dataset involves 237 papers from 24 fields of natural language processing, organized hierarchically with contribution sentences, info units, entities, and

\*equal contribution

Objects to Identify	Examples
Sentence	We use the BERT <sub>BASE</sub> model pre-trained on English Wikipedia and BooksCorpus for 1M steps.
Information Unit	model
Scientific Term and Predicate Phrases	used, BERT <sub>BASE</sub> model, pre-trained on, English Wikipedia, BooksCorpus, for, 1M steps
Triples	(Contribution, has, ExperimentalSetup), (ExperimentalSetup, used, BERT <sub>BASE</sub> model), (BERT <sub>BASE</sub> model, pre-trained on, English Wikipedia), (BERT <sub>BASE</sub> model, pre-trained on, BooksCorpus), (BERT <sub>BASE</sub> model, for, 1M steps)

Table 1: Objects need to be identified

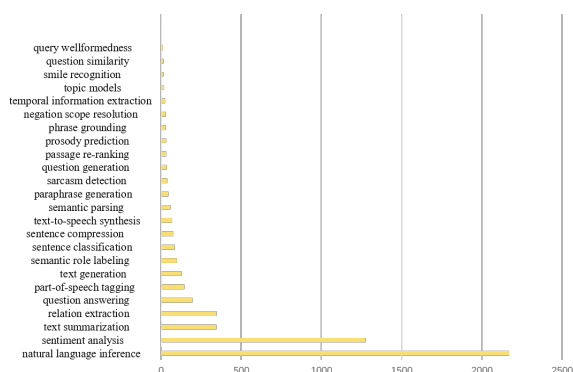


Figure 1: Numbers of annotated sentences in each domain

triples. Thus, for different tasks we can use disparate parts of the dataset.

Inherent challenges also come with data collection. As shown in Fig 1, first, there is a dramatic discrepancy among the number of annotated papers. The NaturalLanguageInference field received the richest resources. A total of over one hundred papers in this area are annotated. On the other hand, for the domains with poor annotation like PhraseGrounding and QueryWellformedness, only a single paper is provided. Moreover, a postdoctoral researcher with a background in natural language processing is responsible for finishing the pilot annotation task (D’Souza and Auer, 2020). Therefore, the annotated data is relatively subjective and sometimes even inconsistent. For example, some information units nested in Experiments actually also included a combination of ExperimentalSetup and Results. Alternatively, it can be combination of Tasks and their Results.

## 2.2 Related Work

A vast amount of excellent work has been done in the areas of these subtasks. Early work employing CNNs, RNNs and attention based RNN or CNN models has made great progress in sentence classification tasks. (Yang et al., 2017; Liu and Zhang, 2017). Tai et al. (2015) inspired innovation in traditional LSTM networks. The tree-LSTM structure mentioned in their paper is enhanced with dependency or constituency trees. Since Graph Neural Network (GNN) is first used for sentence classification tasks, GNN has been one of the most prevalent encoders for Natural Language Processing(NLP) tasks. Transformer based models are also popular encoders. They are so powerful that they have even been widely used in computational vision areas (Dosovitskiy et al., 2020).

The sequence labeling task is a critical component of NLP applications. There are two basic approaches. In the token level approach, a sequence of tokens is used as an input of sequence tagging models, and tags for each token can be output. Other approaches attempt to solve problem on the sentence level. Lu and Roth (2015) designed a hypergraph, which provides a resolution for the discontinuous terms.

Traditional open information extractors are based on rules and statistical approaches, like Stanford-IE(Angeli et al., 2015),OpenIE-5(Saha and Mausam, 2018) and MinIE (Gashteovski et al., 2017).These methods apply semantic parsers combined with predefined rules to extract triples. Recently, neural OpenIE methods dominate this research field. RnnOIE (Stanovsky et al., 2018) inspired by the sequence labeling systems identifies relation phrases first then combine relations with

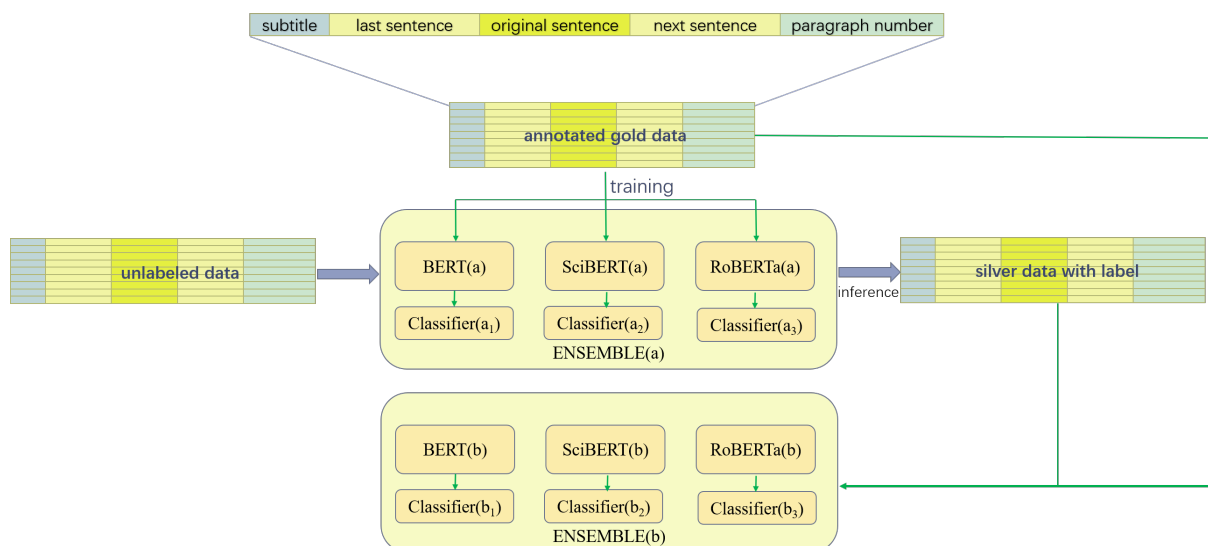


Figure 2: An overview of the system for sentence classification

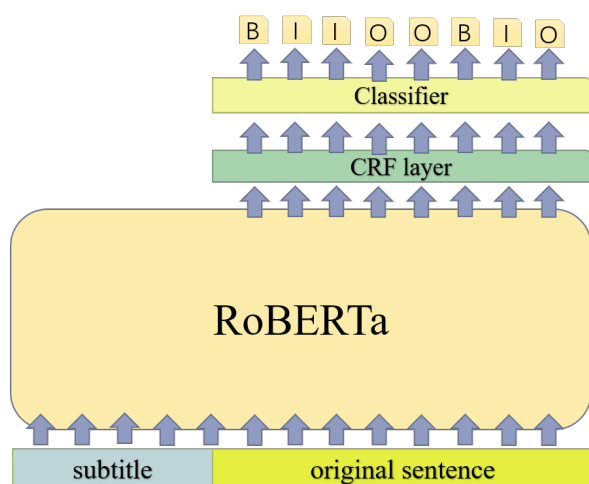


Figure 3: An overview of the system for scientific term extraction

arguments. IMOJIE (Kolluru et al., 2020) takes advantage of seq2seq architectures. It is trained on training data bootstrapped from extractions of several tradition systems such as Stanford-IE.

### 3 System Description

Systems applied for contribution sentence detection and scientific term extraction are based on the RoBERTa (Liu et al., 2019), SciBERT (Beltagy et al., 2019) and basic BERT model with task-specific modifications. For triple classification tasks, a SciBERT (Beltagy et al., 2019) based model is used for candidate triple classification. Moreover, self-training, ensemble and rule design enhanced system performance in different ways.

### 3.1 Contribution Sentence Detection

The contribution sentence detection task is handled as a sentence classification (SC) problem. Let  $U$  be the union of a predefined sentence type set and  $\epsilon$  indicate that the sentence is not a contribution sentence. According to the task description provided by D'Souza et al. (2021), one contribution sentence could belong to one of eleven categories, called info units. Hence  $U$  has twelve elements with  $\epsilon$  added. As shown in Fig2, the input data consisted of four parts: the original sentence, contextual information, a sub-title of the paragraph and the number of paragraph, with the separator token ([SEP]) in between. For contextual information, we used the adjacent sentences of the original one. We define the sub-title of a paragraph as the nearest title found previous to the begin of this paragraph. Besides, the paragraphs are numbered from zero following an increasing order. We add [CLS] token at the top of the sequence and build a classifier on top of its embedding, which is generated from BERT based model, similar to what Devlin et al. (2019) did for pre-training.

Inspired by incremental semi-supervised training (Rosenberg et al., 2005), we introduced the similar training process. Prior to that, we need to prepare the additional unlabeled data. Newest papers are downloaded according to the areas then transformed into Stanza version as the papers provided in training data. The amount of additional data for every field is about ten articles. Training process takes the following steps: first, train BERT based models that will be used in ensemble on gold

sentence	triples
We also apply 3 dense blocks based on char - ResNet which we refer to as char - DenseNet, to compare the difference between residual connection and dense connection.	(Baselines, apply, 3 dense blocks) (3 dense blocks, based on, char - ResNet) (char - ResNet, refer to as, char - DenseNet)

Table 2: A sentence and triples from it

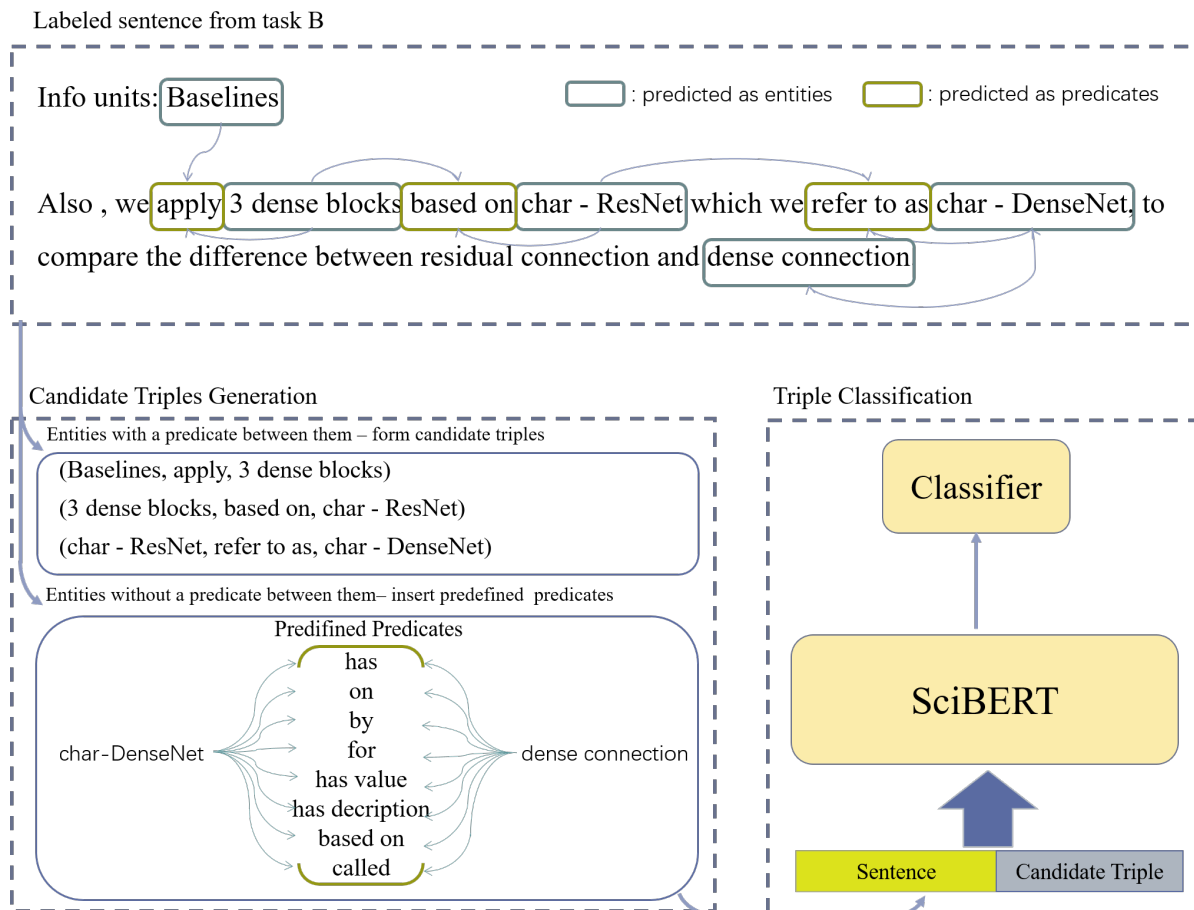


Figure 4: Model overview for triple generation

data, that is the annotated data in dataset, until parameters converged. Next, put these models in an ensemble to tag the unlabelled data for areas suffered from data insufficiency. In the rest part of this passage, we call data labeled by our models 'silver samples' or 'silver data'. Only the sentences all models in the ensemble labelled unanimously would be deemed as silver samples and used for further training. After that, a combination of gold and silver data is input for training another three models with unchanged hyperparameters. Above progress iterates until no further progress could be achieved. That means the iteration is stopped when loss becomes stable. Thus, three fine-tuned BERT based models are ready for inferring and we col-

lect them in an ensemble. When inferring, with a sentence input, each model outputs a vector  $V$ , of twelve dimensions. We calculate a weighted sum of three vectors and use the index of the max element to determine the final class of a sentence. Weights here are hyperparameters and chosen manually.

$$V = 0.25 * V_{SciBERT} + 0.2 * V_{RoBERTa} + 0.1 * V_{BERT}$$

### 3.2 Scientific Term Extraction

We consider scientific term extraction as a sequence labeling (SL) task. Specifically, as Fig 3 shows, a RoBERTa exploiting CRF layer marks every sub-word token of the input sentence with one of the label in B, I, O, where 'B' is the beginning of a term, 'I' indicates that the token is inside the term and

'O' indicates that the token stands outside of the term. In this way, one sequence of a 'B' followed by a continuing sequence of 'I' is recognized as a legal term. Input sequences are also engineered. The sub-title information is attached to the raw sentence. To facilitate triple generation, two models are trained for labeling predicate and entity individually. Discrimination of types of terms allows the system to exploit the information of a predicate more efficiently.

### 3.3 Triple Generation

Given the fact that the outcomes of supervised open information extraction architectures are not as expected, we make use of the result of term extraction. As shown in Table 2, triples from a sentence always overlap at the head and tail. Given that fact, when generating a candidate triple, a predicate acts as the anchor and its neighboring entities are seen as the subject and object, following the sentence reading order, as demonstrated in Fig4. It is rare that the subject appeared later than the object in a sentence.

At times, as task description paper (D'Souza et al., 2021) mentioned, when no suitable predicate phrases could be inferred from the sentence, one candidate from a pre-defined set of predicates could be utilized. The set including "has", "on", "by", "for", "has value", "has description", "based on", "called". We call these triples with predefined predicates "special triples". A greedy matching is introduced that each predefined predicate is inserted between every adjacent entity pair to compose a potential special triple. To illustrate, take the entity pair "char-DenseNet" and "dense connection" as an instance. In Fig4, as the result of term extraction shows that there is no phrase labelled as predicate between "char-DenseNet" and "dense connection". To form candidate triples for this entity pair, each predicate in the predefined predicate set is inserted between the entity pair. After all potential triples are generated, gather unions of each candidate triple and the sentence where the triple came from as input data. A SciBERT based binary classifier then judge if a union is rational. We refer to it as the 'candidate triple judge model' in the rest of this article.

Additionally, another rule is designed for cross-sentence triples, which takes up three percent of all triples. Such amount cannot be ignored also. We observe that when only one term can be extracted from a sentence, it is highly possible that the term

Hparam	SC	TE	TG
Number of epochs	8	20	10
Max length	200	128	256
Batch size	32	16	32
Learning rate	2e-5	1e-5	1e-5
Optimizer	AdamW		
Loss	cross entropy		

Table 3: Hyperparameters for models. SC means sentence classification, TE stands for term extraction and TG is the abbreviation of triple generation

is a composition of a cross-sentence triple. Such terms adjacent to each other are integrated into a cross-sentence triple according to the subject-predicate-object order. From example, if there are three adjacent contribution sentences that we can only extract one phrase from each, and these three phrases are predicted as entity, predicate and entity respectively. Thus we can combine them together as a cross-sentence triple. For these triples, we do not apply a further filter and add them into the final output directly.

## 4 Experimental Setup

In this section, we describe the models we used in the final submission and their parameters in detail. It should be possible to reproduce our work.

### 4.1 Models and Parameters

Before training, we divide papers into three groups: train set, dev set, and test set, with a ratio of 8:1:1. We then mix all sentences in each set together. In this way, data leakage is prevented. Otherwise, sentences in test set and train set could come from the same article. The hyperparameters for training are shown in Table3. Our implementation uses only Pytorch for the first two sub-tasks' models and AllenNLP for the last candidate triple judge model.

For Contribution Sentence Classification task, we attempt to take advantage of diverse models. An ensemble of BERT, SciBERT and RoBERTa is applied. During the training process, F1 score on dev set works as a criteria for choosing the best epoch and model weights. Additionally, because our model is consistently confused between the info units of Approach and Model, we convert sentences with the word 'approach' to the unit Approach after receiving predictions from a neural network. For Scientific Term Extraction, we used

		<b>Contribution Sentence Detection</b>	
		sentence detection	information unit classification
<b>F1 of ensemble</b>		0.3978	0.8108
<b>F1 of SciBERT</b>		0.3856	0.8049

		<b>Scientific Term Extraction</b>		<b>Triple Generation</b>	
		RoBERTa+CRF+BIO	RoBERTa+span tagging	rule based method	IMOJIE
<b>F1</b>		0.7774	0.7567	0.4473	0.1729

Table 4: F1 scores of models. For the submitted model, the F1 scores are from the leader board , for the baseline model the F1 scores are from results on dev set

RoBERTa as the encoder and elaborated more on different decoders. The basic BIO tagging model performs better than the span based one. When training the potential triple judge model, the learning rate rises first then falls following the method used by Vaswani et al. (2017)

## 4.2 Baselines

We endeavor to search for the best baselines. For sentence classification, we use single SciBERT model as our baseline, while for sequence tagging, we employ RoBERTa without CRF layer and rather using span tagging decoder as a strong baseline. In the triple generation task, we once tried to employ neural open information extraction models, so IMOJIE can be deemed as a baseline.

## 4.3 Evaluation Metric

We use F1 value as the main metric, the average F1 is arithmetic mean value of sentence F1, terms F1, info units F1, and triples F1. When computing triples F1, strict standard is employed. Only when every division of a predicted triple matched the gold answer, it can be counted as a correct inference.

$$F1 = \frac{2 * P * R}{P + R} \quad (1)$$

where  $P$  means precision of the prediction and  $R$  means recall.

$$F1_{avg} = avg(F1_{sentence}, F1_{terms}, F1_{infounits}, F1_{triples}) \quad (2)$$

## 5 Results

With the gold data of the upstream task provided, the F1 value of sentence, info units, terms, and triples are 0.3978, 0.8108, 0.7774, and 0.4473 respectively, as shown in Table4 .

The enhancement in the sentence classification is clear. As a prevalent technology, ensemble has become a necessary part of algorithm competitions.

The only restriction is that all models in ensemble should have an F1 over fifty percent. With the help of data augmentation, the info unit classification result occupied 2<sup>nd</sup> position in the final ranking.

It marvels us that the model with span based tagger performed worse than the BIO tagger. Many NER experiments shows the evidence that the span based tagging decoder outperforms the simple BIO tagger. We believe the main reason is that, when exposed to the data in this task, there is no need to discern types of entities. While the span based decoder are equipped with the ability to infer entity types, such design may be suboptimal and create additional errors.

To some extent, the improvements in the triple generation task proves that neural OIE models are inapplicable to the task on this dataset. The main cause may be the different definitions of 'predicate.' In our task, prepositions always appear in the position of predicate in triples. Likewise, the subject and object are persons or specific terms while for sentences in science papers only scientific notions can be found. Given so much elaboration in our system, terms extraction and triple generation task also achieved the second place on the leaderboard.

## 6 Conclusion

We engaged in SemEval-2021 task11 NLPContributionGraph with models integrating features suitable for disparate tasks. We took insight on the impact of different parts on the final results, fine-tuned hyperparameters, and attempted various feature engineering methods. We ranked 4<sup>th</sup>, 2<sup>nd</sup>, and 2<sup>nd</sup> in three evaluation phases and our final model demonstrated its superiority over several strong baselines.

## Acknowledgments

The task is sponsored by ICA group, East China Normal University. Besides, thanks are due to Yi-



wei Yan for her supervision as without her help it would have been impossible to finish writing so quickly.

## References

- Gabor Angeli, M. Johnson, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.
- Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. [SciBERT: A pretrained language model for scientific text](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [BERT: Pre-training of deep bidirectional transformers for language understanding](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. [An image is worth 16x16 words: Transformers for image recognition at scale](#).
- Jennifer D’Souza, Sören Auer, and Ted Pedersen. 2021. [SemEval-2021 task 11: Nlpcontributiongraph - structuring scholarly nlp contributions for a research knowledge graph](#). In *Proceedings of the Fifteenth Workshop on Semantic Evaluation*, Bangkok (online). Association for Computational Linguistics.
- Jennifer D’Souza and Sören Auer. 2020. [Nlpcontributions: An annotation scheme for machine reading of scholarly contributions in natural language processing literature](#).
- Kiril Gashteovski, Rainer Gemulla, and Luciano del Corro. 2017. [MinIE: Minimizing facts in open information extraction](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2630–2640, Copenhagen, Denmark. Association for Computational Linguistics.
- Mohamad Yaser Jaradeh, Allard Oelen, Kheir Ed-dine Farfar, Manuel Prinz, Jennifer D’Souza, Gábor Kismihók, Markus Stocker, and Sören Auer. 2019. *Open Research Knowledge Graph: Next Generation Infrastructure for Semantic Scholarly Knowledge*, page 243–246.
- Keshav Kolluru, Samarth Aggarwal, Vipul Rathore, Mausam, and Soumen Chakrabarti. 2020. [IMoJIE: Iterative memory-based joint open information extraction](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5871–5886, Online. Association for Computational Linguistics.
- Jiangming Liu and Yue Zhang. 2017. [Attention modeling for targeted sentiment](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 572–577, Valencia, Spain. Association for Computational Linguistics.
- Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. [Roberta: A robustly optimized bert pretraining approach](#). *ArXiv*, abs/1907.11692.
- Wei Lu and Dan Roth. 2015. [Joint mention extraction and classification with mention hypergraphs](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal. Association for Computational Linguistics.
- C. Rosenberg, M. Hebert, and H. Schneiderman. 2005. [Semi-supervised self-training of object detection models](#). In *2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION’05) - Volume 1*, volume 1, pages 29–36.
- Swarnadeep Saha and Mausam. 2018. [Open information extraction from conjunctive sentences](#). In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.
- Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. [Supervised open information extraction](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895, New Orleans, Louisiana. Association for Computational Linguistics.
- Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. [Improved semantic representations from tree-structured long short-term memory networks](#). In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).

Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and  
Xiaojun Chen. 2017. [Attention based lstm for target  
dependent sentiment classification.](#)

# UoR at SemEval-2021 Task 12: On Crowd Annotations; Learning with Disagreements to Optimise Crowd Truth

Emmanuel Osei-Brefo, Thanet Markchom , Huizhi Liang

University of Reading

White Knights, Berkshire, RG6 6AH

United Kingdom

e.osei-brefo@pgr.reading.ac.uk, t.markchom@pgr.reading.ac.uk

huizhi.liang@reading.ac.uk

## Abstract

Crowdsourcing has been ubiquitously used for annotating enormous collections of data. However, the major obstacles to using crowd-sourced labels are noise and errors from non-expert annotations. In this work, two approaches dealing with the noise and errors in crowd-sourced labels are proposed. The first approach uses Sharpness-Aware Minimization (SAM), an optimization technique robust to noisy labels. The other approach leverages a neural network layer called softmax-Crowdlayer specifically designed to learn from crowd-sourced annotations. According to the results, the proposed approaches can improve the performance of the Wide Residual Network model and Multi-layer Perception model applied on crowd-sourced datasets in the image processing domain. It also has similar and comparable results with the majority voting technique when applied to the sequential data domain whereby the Bidirectional Encoder Representations from Transformers (BERT) is used as the base model in both instances.

## 1 Introduction

In recent years, there has been some major advancement in the use of deep learning for solving artificial intelligence problems in different domains such as sentiment analysis, image classification, natural language inference, speech recognition object detection. They have also been used in many other numerous cases where human disagreements are encountered such as speech recognition, visual object recognition, object detection and machine translation (Rodrigues and Pereira, 2018). It is however, an essential requirement for deep learning models to utilise labelled data to undertake the representational learning of the underlying datasets. These labelled data are most at times not available and hence the need for humans to manually undertake the labelling of these data becomes a necessity.

In recent years, crowd-sourcing has been used in the annotation of large collections of data and has proven to be an efficient and cost-effective means of obtaining labeled data as compared to expert labelling (Snow et al., 2008)

It has been utilised in the generation of image annotations to train computer vision systems (Raykar et al., 2010), to provide the linguistic annotations used for Natural Language Processing (NLP) tasks (Snow et al., 2008), and has also been used to collect the relevant judgments needed to optimize search engines (Alonso, 2013).

It is a well known fact that crowd-sourced labels are known to be associated with noise and errors as a result of the annotations being provided by annotators with uneven expertise and dedication which can result in the compromise of practical applications that uses such data (Zhang et al., 2016). This paper therefore seeks to apply a novel approach to minimize and mitigate the noise and errors in crowd sourced labels. The aim is to investigate the use of a unified testing framework to learn from disagreements using crowd source labels collected from different annotators.

## 2 Related Work

Crowdsourcing has proven to be an inexpensive and efficient way to collect large labels of data and has attracted much research interest from the machine learning community to address noise and unreliabilities associated with them. The proposal for using an Expected Maximization (EM) algorithm to obtain density estimate rate of errors of patients providing conflicting responses to medical questions by Dawid and Skene (1979), is one of the key pioneer contributions to this field. This work served as the catalyst for many other approaches used for the aggregation of labels from crowd annotators with different levels of expertise, such as

the one proposed in [Whitehill et al. \(2009\)](#), which further extends Dawid and Skene’s model by also accounting for item difficulty in the context of image classification. Similarly, [Ipeirotis et al. \(2010\)](#) proposed using Dawid and Skene’s approach to extract a single quality score for each worker that low-quality workers to be pruned. The approach proposed in our paper contrast with this line of work, by allowing neural networks to be trained directly on the softmax output of the noisy labels of multiple annotators, thereby avoiding the need to resort to prior label aggregation schemes. [Smyth et al. \(1995\)](#) also collated the opinions of many experts to establish ground truth and there has been a large body of research work using EM approaches to annotate labels for datasets by many experts ([Whitehill et al., 2009](#); [Raykar and Yu, 2012](#)).

[Rodrigues et al. \(2014\)](#) also used the EM approach of labelling datasets by experts through the use of Gaussian Process classifiers. [Rodrigues and Pereira \(2018\)](#) also deployed the use of crowd layer with a CNN model to capture the biases of different annotators and correct them, our approach is the first to be built on the Wide Residual Network (WideResNet) model ([Zagoruyko and Komodakis, 2017](#)) and the Bidirectional Encoder Representations from Transformers (BERT) model ([Devlin et al., 2019](#)). Our approach differs from the method used by [Rodrigues and Pereira \(2018\)](#) because our technique initially finds the softmax of the output of the crowd responses before it is used for the modelling whereas the [Rodrigues and Pereira \(2018\)](#) approach works on the responses from the crowd directly.

### 3 Systems Description

These systems are proposed for image classification tasks and NLP tasks with sub-task-specific modifications and training schemes applied to each of the dataset.

#### 3.1 softmax-Crowdlayer

A special type of network layer known as softmax-Crowdlayer initially proposed by ([Rodrigues and Pereira, 2018](#)), was used to train a deep neural network directly from the noisy labels of multiple annotators from the crowd-sourced data. It used the output layer of a deep neural network as its input and was trained to learn from an annotator-specific mapping from the output layer to the labels of the different soft-maxed crowd annotators; and by so

doing it was able to learn the reliability and biases of each annotator in the process. As can be seen from [Figure 1](#), which is the generalised architecture encompassing either a Multi-layer Perceptron (MLP), WideResNet, or BERT as its’ base model, was used together with a softmax-Crowdlayer for the respective datasets. The output layer from the deep neural network served as a bottleneck and input for the crowd Annotators to learn from. It used a specialised cross-entropy loss known as Masked Multi Cross Entropy loss during training to handle the missing answers from Annotators. After the training of the network with the crowd layer and the specialised loss function, the crowd layer was removed to expose the Bottleneck layer which was then used to make the predictions.

The intuition behind the deployment of the crowd layer on top of the base model was that; the softmax-Crowdlayer would adjust the gradients from the labels of each annotator depending on their level of expertism and adjusts their weights and propagate the errors through the entire neural network system.

Sections [3.2](#) covers the use of WideResNet together with SAM on the CIFAR10-IC dataset whilst sections [3.3](#) and [3.4](#) covers the use of softmax-Crowdlayer for image classification whilst section [3.5](#) explores the use of BERT and softmax-Crowdlayer to cover the NLP aspect of the task which has been visualised in [figure 1](#). The motivation behind the preference of the BERT model over the baseline models was to investigate the potential of using BERT, which is a state-of-the-art model, with the softmax-Crowdlayer.

#### 3.2 WideResNet with Sharpness Aware Minimisation (SAM) For Majority Voting

The CIFAR10 dataset had a model made with WideResNet; first implemented by ([Zagoruyko and Komodakis, 2017](#)). A widening factor of 12 and convolutions of size together with 16 layers were used. A learning rate of 0.1 with weight decay of 0.001 and momentum of 0.8 was used with the SAM optimiser which had stochastic Gradient Descent (SGD) as its base optimiser. The training epochs for the dataset were scheduled in batches of 1000 for 60, 5, 10 and 20 respectively. The minimization of the commonly used loss functions such cross-entropy and the use of the custom Masked loss function designed specifically for the crowd layer on the CIFAR10-IC were not sufficient to

achieve superior results since the training loss landscapes of models used for noisy labels are complex and non-convex, with a multiplicity of local and global minima (Foret et al., 2020).

The Sharpness-Aware Minimization (SAM) Foret et al. (2020), was applied to the CIFAR dataset with the use of WideResNet model generalization which aided in the simultaneous loss in value and sharpness of the noisy labels from the crowd annotators as it has been shown to be robust to noisy labels (Foret et al., 2020). The inner working of the sharpness Aware Minimization is such that rather than using a parameter value that simply have low training loss value, a parameter value whose entire neighborhoods have uniform training loss value is the utilised.

The SAM optimiser technique was not applied to the NLP tasks because, its performance on them was not as good as that of the CIFAR-10 dataset.

### 3.3 WideResNet with softmax-Crowdlayer for CIFAR10-IC Dataset

The CIFAR10-IC data was made up of transformed Images that belonged to one of the 10 classes below: ‘plane’, ‘car’, ‘bird’, ‘cat’, ‘deer’, ‘dog’, ‘frog’, ‘horse’, ‘ship’, ‘truck’.

The WideResNet described in section 3.2 was used as the base model which had a softmax-Crowdlayer added to the output layer and through the action of back-propagation, it was able to correct the errors of the 2571 Annotators. A training epoch of 400 and batch size of 64 were used for with this approach. One hot encoding, together with a specialised function were used to generate the set of missing annotations which was then trained using the masked multi cross-entropy loss function for error corrections and predictions through the weights update.

### 3.4 MLP with softmax-Crowdlayer for LabelMe-IC Dataset

The LabelMe-IC data was made up of VGG16 encoded images that belonged to one of the 8 categories or classes below: ‘highway’, ‘inside city’, ‘tall building’, ‘street’, ‘forest’, ‘coast’, ‘mountain’ or ‘open country’

This was an image classification task that had a standard MLP architecture together with softmax-Crowdlayer applied to it. The MLP was made up of 4 hidden layers with 128 Relu Units each, an optimiser made of Adam optimizer, loss function made of categorical cross entropy and a drop out

of 0.2. A training epoch of 400 and batch size of 32 were used. The output layer had a softmax activation that outputted to the 8 distinct classes highlighted earlier. The softmax-Crowdlayer described in section 3.1 was then connected to this output layer where the Annotators errors and biases were back-propagated through a training scheme which reduced the noise in the crowd Annotators through the use of a specialised loss function to handle crowd annotations known as masked multi cross entropy loss function.

### 3.5 BERT with softmax-Crowdlayer for Gimpel-POS and PDIS Datasets

In Gimpel-POS dataset, each sample consisted of a tweeted text, a specific word/token appears in a tweeted text and a crowd label which is a list of multiple labels from different annotators. The task was to predict a part of speech (POS) of a given token. The POS labels include ‘ADJ’ (adjective), ‘ADP’ (adposition), ‘ADV’ (adverb), ‘CCONJ’ (coordinating conjunction), ‘DET’ (determiner), ‘NOUN’ (noun), ‘NUM’ (numeral), ‘PRON’ (pronoun), ‘PART’ (particle or other functional word), ‘PUNCT’ (punctuation), ‘VERB’ (verb) and ‘X’ (others). Table 1 shows an example of Gimpel-POS dataset. In this example, ‘Texas’ is a token needed to be tagged. It is at the beginning of the tweeted text shown in the first row. Considering the crowd label provided, the first and the second annotators both labeled this token as a noun, while the last annotator labeled this token as a pronoun.

Tweeted text	Texas Rangers are in the World Series! Go Rangers!
Token	Texas
Crowd label	[NOUN,NOUN,PRON]

Table 1: An example of Gimpel-POS dataset

Considering PDIS dataset, the goal was to predict whether a given noun phrase refers to new information or to old information in a document. Each sample consisted of a document (tokenised sentences), a noun phrase appear in the document, a pre-computed syntactic feature of a given noun phrase, and a crowd label. Table 2 shows an example of PDIS dataset. The document and the noun phrase are in the first and the second row of the table respectively. The noun phrase is ‘The cat’ at the beginning of the document. Syntactic feature

of this noun phrase is a feature vector shown in the third row. The fourth row shows a crowd label of the given noun phrase. The first and the second annotator labeled the noun phrase as 0 and 1 respectively 0 means that the noun phrase refers to new information and 1 means that it refers to old information.

Document	The cat ate the rat. Thereafter the dog ate the cat.
Noun phrase	The cat
Syntactic feature	[0,1,0,...,0]
Crowd label	[0,1]

Table 2: An example of PDIS dataset

In this work, we propose to fine-tune the pre-trained BERT model for both Gimpel-POS task and PDIS task based on crowd labels. To do so, the original input format of both tasks was firstly converted to the BERT conventional format. For each sample in Gimpel-POS dataset, a tweeted text and a given token were first concatenated in the following format:

[CLS] Tweeted text [SEP] Token [SEP]

where ‘[CLS]’ token is added for classification and two ‘[SEP]’ tokens are used to identify the boundary of a tweeted text and a token. Similarly, for PDIS dataset, a document is also concatenated with a noun phrase as follows:

[CLS] + Document + [SEP] + Noun phrase + [SEP]

These concatenated texts are used for fine-tuning the pre-trained BERT model.

To fine-tune the pre-trained BERT model, a dense layer was added at the end of the pre-trained BERT model. This layer took a ‘[CLS]’ token embedding from the pre-trained BERT model as an input and outputted a vector with the size equal to the number of classes in either dataset (12 for Gimpel-POS and 2 for PDIS). A softmax activation layer was added after the dense layer to compute the probabilities of each class. These additional layers can be seen as a classifier module that is added on top of the pre-trained BERT model. This is a common way to fine-tune the pre-trained BERT model for a specific task with regular labels as targets (Devlin et al., 2019).

In order to deal with crowd labels in the datasets, the softmax-Crowdlayer was added next to the classifier module. Similarly to the MLP model with

the crowd layer highlighted in 3.4, The proposed model for fine-tuning the pre-trained BERT with the softmax-Crowdlayer is illustrated in Figure 1. The Gimpel-POS example in 1 is used for demonstration in this figure. As previously mentioned, only the ‘[CLS]’ token is passed through the additional classifier module to predict primary classification output. This output is further used as an input of the softmax-Crowdlayer to predict the final output as described in the previous section. The proposed model can be instantly applied with PDIS dataset by changing the output size of the dense layer in the classifier module to 2. Due to lack of resources, the fine tuning of all the Bert model was run for 1 epoch.

## 4 Results and Discussion

The results were evaluated using two metrics known as  $F1$  score, referred to as hard evaluation and cross Entropy, referred to as soft evaluation. Models with Higher  $F1$  scores and lower cross entropy values are the desired outcomes expected from the models.

As can be seen in Table 3, The use of MLP together with the softmax-Crowdlayer on the LabelMe-IC dataset achieved the highest  $F1$  score of 0.7839, which was 0.739 greater than the majority voting model provided as the baseline model by the task organisers and also had a comparative lowest cross entropy value of 1.7693. The vast difference in the performance of the majority voting and the softmax-Crowdlayer can be attributed to the calculation of the number of missing annotations together with the ability of the softmax-Crowdlayer to learn the true labels from the crowd labels. This leads to the correction of the errors and mislabelling from inexperienced annotators through the process of back-propagation. The majority voting does not have this unique ability and therefore uses the wrong labelling without any of such adjustments.

The use of WideResNet together with SAM resulted in a superior performance with  $F1$  score of 0.7693 and cross entropy of 0.8274 as compared to the performance WideResnet with the softmax-Crowdlayer which had an  $F1$  score of 0.4427 and cross entropy of 1.9286 when applied to the CIFAR10-IC. It’s cross entropy of 1.9286 was better than the baseline majority method which was 2.8306. The PDIS data which was fine-tuned with a pre-trained BERT model plus softmax-crowdlayer

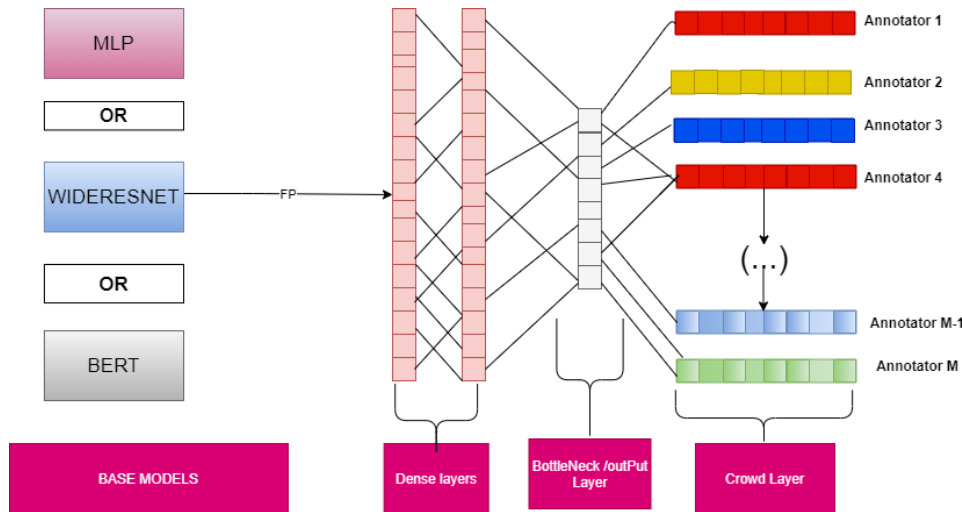


Figure 1: The proposed softmax-Crowdlayer on top of the respective Base Models

had  $F1$  score of 0.4379 and cross entropy of 0.8295. The BERT + softmax-Crowdlayer did not perform comparatively well when applied to the Gimpel-POS data since it only managed to achieve an  $F1$  score of 0.1254 and corresponding cross entropy of 2.3318. From Table 3 it can also be seen that the BERT + Majority voting had the same results as the BERT + softmax-Crowdlayer model so further investigation needs to be conducted to find out why this was so. As can be seen in Table 3, the use of the full base model provided for the PDIS and Gimpel-POS by the organisers achieved superior results and should have been used with the softmax-crowdlayer, but it could not be done because the full base model provided by the organisers was written in Pytorch framework whilst the softmax-crowdlayer was written in Keras. There should therefore have been the need to convert the full base model to Keras before using the softmax-crowdlayer and it's eventual evaluation, but as a result of the limited availability of time, it has been reserved as part of our future work to be covered in section 5

Refer to Appendix A for the analysis of the class distribution of the datasets.

## 5 Conclusion

This paper used a softmax-Crowdlayer approach combined with a deep neural network to train noisy labels from multiple crowd annotators. WideResNet together with softmax-Crowdlayer has been applied on CIFAR10-IC datasets, whilst MLP combined with softmax-Crowdlayer has been used on the LabelMe-IC data and BERT combined with

softmax-Crowdlayer has been used on Gimpel-POS and PDIS data respectively.

Future work will explore the effect of the distribution of the class annotation on the labeling accuracy and also investigate more efficient approaches of combining the BERT model with the softmax-Crowdlayer to further improve the results. It will also involve the application of the softmax-crowdlayer on the Humour dataset which was not included in this work due to time constraint posed as a result of the complicated data points of the humour dataset.

## References

- Omar Alonso. 2013. [Implementing crowdsourcing-based relevance experimentation: an industrial perspective](#). *Information Retrieval*, 16(2):101–120.
- A. P. Dawid and A. M. Skene. 1979. [Maximum likelihood estimation of observer error-rates using the em algorithm](#). *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. [Sharpness-aware minimization for efficiently improving generalization](#).
- Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. [Quality management on amazon mechanical turk](#). In *Proceedings of the Acm sigkdd Workshop on Human Computation*, hcomp '10, page 64–67, New York, NY, USA. Association for Computing Machinery.

SubTask	Model	F1 Score	Cross Entropy
CIFAR10-IC	WideResNet + Majority Voting+SAM	<b>0.7693</b>	<b>0.8274</b>
CIFAR10-IC	WideResNet + Majority Voting (NO SAM)	0.7156	1.1163
CIFAR10-IC	WideResNet + softmax-Crowdlayer	0.4427	1.9286
CIFAR10-IC	ResNet + Majority Voting	0.6306	2.8306
LabelMe-IC	MLP + softmax-Crowdlayer	<b>0.7839</b>	<b>1.7693</b>
LabelMe-IC	MLP + Majority Voting	0.0449	2.2100
PDIS	BERT + softmax-Crowdlayer	0.4739	0.8295
PDIS	BERT + Majority Voting	0.4739	0.6987
PDIS	CharCNN + Majority Voting	0.5611	0.6892
Gimpel-POS	BERT + softmax-Crowdlayer	0.1254	2.3318
Gimpel-POS	BERT + Majority Voting	0.1254	2.257
Gimpel-POS	RNN + Majority Voting	0.7626	1.0884

Table 3: Evaluation results

Vikas C. Raykar and Shipeng Yu. 2012. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *J. Mach. Learn. Res.*, 13(null):491–518.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. [Learning from crowds](#). *Journal of Machine Learning Research*, 11(43):1297–1322.

Filipe Rodrigues and Francisco Pereira. 2018. [Deep learning from crowds](#).

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2014. [Gaussian process classification and active learning with multiple annotators](#). In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 433–441, Beijing, China. PMLR.

Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. 1995. [Inferring ground truth from subjective labelling of venus images](#). In *Advances in Neural Information Processing Systems*, volume 7. MIT Press.

Rion Snow, Brendan O’Connor, Daniel Jurafsky, and Andrew Ng. 2008. [Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks](#). In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, Honolulu, Hawaii. Association for Computational Linguistics.

Jacob Whitehill, Ting-fan Wu, Jacob Bergsma, Javier Movellan, and Paul Ruvolo. 2009. [Whose vote should count more: Optimal integration of labels from labelers of unknown expertise](#). In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc.

Sergey Zagoruyko and Nikos Komodakis. 2017. [Wide residual networks](#).

J. Zhang, X. Wu, and V. Sheng. 2016. Learning from crowdsourced labeled data: a survey. *Artificial Intelligence Review*, 46:543–576.

## A Class label distribution analysis

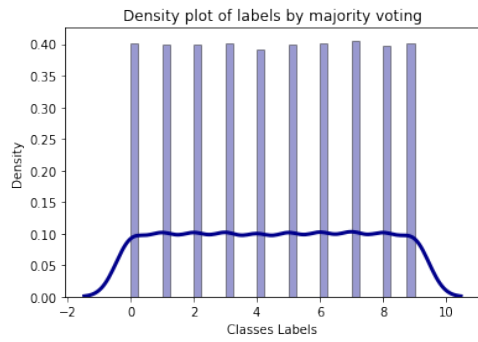
The table 4, summarises the number of Annotators, number of data points, and number classes for each data set used. The Figure 2 contains the probability density estimates of how the annotators perceived the class labels to which each respective item belonged to. Based on the simple majority voting, it can be observed from Figure 2(a) that the distribution was uniform across all classes with the labeling ; **[0:airplane, 1:automobile, 2:bird, 3:cat, 4:deer, 5:dog, 6:frog, 7:horse, 8:ship, and 9:truck ]** for the CIFAR10-IC dataset.

Figure 2(b) depicting the kernel density estimates of the LabelMe-IC data, captures the distribution of how the annotators labelled the data into their respective classes. Majority of the samples were labelled as forest with the respective encoding of the labels shown as; **[0:highway, 1:inside city, 2:tall building, 3:street, 4:forest, 5:coast, 6:mountain, 7:open country]**.

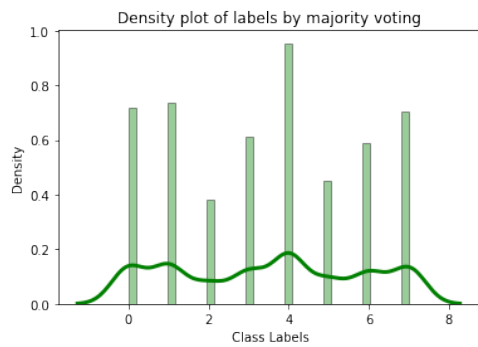
The distribution of the Gimpel-POS and PDIS datasets are represented in figures 2(c) and (d) respectively whose encoded labels have been provided earlier in section 3.5. The encoding for the Gimpel-POS class is shown as **[0:ADJ, 1:ADP, 2:ADV, 3:CCONJ, 4:DET, 5:NOUN, 6:NUM, 7:PRON, 8:PART, 9:PUNCT, 10:VERB, 11:X]**.

The labels of PDIS dataset are encoded as **[0: refer to new information, 1: refer to old information]**.

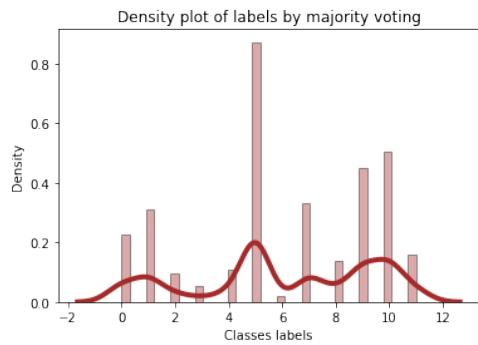




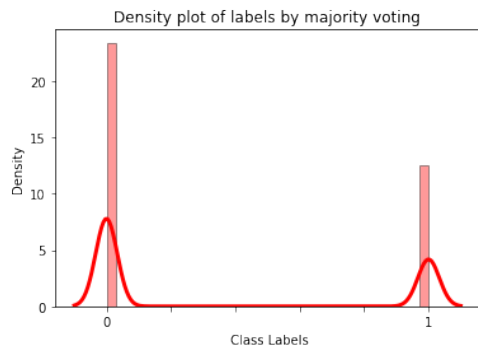
(a)



(b)



(c)



(d)

Dataset	#annotators	#classes	size
CIFAR10-IC	2571	10	7000
LabelMe-IC	59	8	5000
Gimpel-POS	177	12	8310
PDIS	1728	2	86936

Table 4: Summary statistics for each dataset used

Figure 2: The kernel density plot of the distribution of the crowd labels by Annotators for the (a) CIFAR10-IC, (b) LabelMe-IC, (c) Gimpel-POS and (d) PDIS datasets



# Author Index

- Abdullah, Malak, 527, 661  
Abdullah, Malak A., 1068  
AbedulNabi, Isra'a, 1114  
Abu Alasal, Sanaa, 1102  
Abujaber, Dia, 1068  
Acharya, Kaushik, 1271  
Adak, Sayantan, 678  
Agarwal, Raksha, 120  
Akrah, Samuel, 1196  
Aksakalli, Cüneyt, 909  
Al Bashabsheh, Emran, 1102  
Al-Hajj, Moustafa, 748  
Al-Omari, Hani, 1114  
Al-Omari, Sarah, 661  
Al-Sobh, Eslam, 661  
Alam, Firoj, 70  
Alami, Ahmed, 865  
Alami, Hamza, 865  
Almeida, Raul, 683  
Alva-Manchego, Fernando, 144  
Amato, Giuseppe, 1020  
Androustopoulos, Ion, 59  
Andruszkiewicz, Piotr, 974  
Arampatzis, Avi, 1125  
Arefyev, Nikolay, 157, 756, 780, 1249  
Arora, Hardik, 502  
Arya, Karm Veer, 805  
Asklöv, Elin, 632  
Auer, Sören, 364  
Avram, Andrei-Marius, 534  
Ayman, Nabila, 1204  
Aziz, Abdul, 627
- Babaei Giglou, Hamed, 948  
Bagherzadeh, Parsa, 410  
Bahrak, Behnam, 995  
Bai, Yang, 598, 719, 904, 1141  
Bani Yaseen, Tuqa, 661  
Bansal, Archit, 211  
Bao, Hongchang, 763  
Bartsch, Sabine, 130  
Basafa, Hossein, 205  
Bedwell, James, 860  
Beksa, Katarzyna, 974
- Benlahbib, Abdessamad, 865  
Berend, Gábor, 169  
Bergler, Sabine, 404, 410  
Berrada, Ismail, 585, 1135  
Bestgen, Yves, 571  
Bethard, Steven, 348, 458  
Bhartia, Yash, 189, 233  
Bin Ali, Bishr, 70  
Bitinis, Mironas, 632  
Bodnar, Ciprian, 787  
Bordzicka, Zuzanna, 974  
Brookman, Gabriel, 1190  
Bujnowski, Paweł, 974  
Burtenshaw, Ben, 898
- Cao, Jiarun, 1239  
Cech, Maggie, 1003  
Cercel, Dumitru-Clementin, 225, 534, 609, 1160  
Chamberlain, Jon, 338  
Chatterjee, Niladri, 120  
Chaurasia, Sajal, 467  
Chen, Hao, 183  
Chen, Huajun, 810  
Chen, Li, 99  
Chen, Qin, 1295  
Chen, Ruijun, 841  
Chen, Xi, 1239  
Chen, Xiang, 810  
Chen, Xiangnan, 810  
Chen, Xuyi, 286  
Chen, Zhixiang, 827  
Cheng, Fei, 17  
Chersoni, Emmanuele, 565  
Chhablani, Gunjan, 189, 233  
Chi, Ethan A., 688, 1245  
Chi, Gordon, 688  
Chi, Nathan, 1209  
Chi, Ryan, 1209  
Chiruzzo, Luis, 105  
Choe, Enok, 688  
Chy, Abu Nowshed, 627, 990, 1088, 1204  
Cohn, Trevor, 445  
Cook, Paul, 650  
Cox, Jessica, 306

Da San Martino, Giovanni, 70  
Dale, David, 927  
Dang, Yixue, 521  
Daniel Jr., Ron, 306  
Danilevsky, Marina, 317  
Dascalu, Mihai, 225, 534, 609, 1160  
Dash, Tirtharaj, 189  
Davletov, Adis, 780, 1249  
Davletov, Emil, 1249  
Delil, Selman, 909  
Deng, Shumin, 810  
Dernoncourt, Franck, 397  
Desai, Abhinandan Tejalkumar, 548  
Dimitrov, Dimitar, 70  
Ding, Huiyang, 263  
D'Souza, Jennifer, 364  
Dumitrache, Anca, 338  
Duwairi, Rehab, 1114

Ebrahimi, Ali, 205  
Effrosynidis, Dimitrios, 1125  
Eisenschlos, Julian, 423  
El Mahdaouy, Abdelkader, 585, 1135  
El Mamoun, Nabil, 585, 1135  
El Mekki, Abdellah, 585, 1135  
Elkayam, Otniel, 138  
Essefar, Kabil, 585, 1135  
Evans, Richard, 1

Fabro, Marcos Didonet Del, 683  
Faili, Heshaam, 205  
Fakharian, Samin, 650  
Falchi, Fabrizio, 1020  
Fan, Hongjie, 258  
Fan, Xiaoran, 286  
Faraj, Dalya, 527  
Feng, Shikun, 99, 286  
Feng, Yuxi, 183  
Feng, Zhida, 99  
Firlag, Klaudia, 974  
Firooz, Hamed, 70  
Flynn, Robert, 603  
Fornaciari, Tommaso, 338  
Fermann, Lea, 445

Gangwar, Akash, 1232  
García-Díaz, José Antonio, 1096  
Gautam, Devansh, 1075, 1262  
Gennaro, Claudio, 1020  
Ghadery, Erfan, 1015  
Ghosal, Tirthankar, 502  
Ghosh, Sreyan, 249

Gia Hoang, Phu, 919  
Gifu, Daniela, 787, 1226  
Glazkova, Anna, 913  
Goltz, Christian, 974  
Gombert, Sebastian, 130  
Gooch, Phil, 502  
Gordeev, Denis, 780, 1249  
Goyal, Harsh, 743  
Grad, Dawid, 860  
Groth, Paul, 306  
Gu, Jinghang, 565  
Guan, Zhengyi, 1108  
Gugnani, Akshay, 1190  
Gui, Min, 1082  
Guo, Guibing, 827  
Gupta, Aishwarya, 290  
Gupta, Ambuje, 941  
Gupta, Ankur, 327  
Gupta, Aviansh, 1185  
Gupta, Kshitij, 1075, 1262  
Gupta, Priyanshu, 438  
Gupta, Rohan, 511  
Gupta, Vansh, 1061

Haines, Carl, 1179  
Hakimi Parizi, Ali, 650  
Halder, Tanurima, 678  
Hao, Zhenghong, 820  
Harper, Corey, 306  
Hauer, Bradley, 763  
He, Changhong, 485  
He, Liang, 1295  
he, liang, 183  
Hegde, Nidhi, 438  
Hernández, Héctor, 632  
Hettiarachchi, Hansi, 771  
Homan, Christopher, 548  
Hossain, MD. Akram, 627  
Hossain, Tashin, 990, 1088  
Hou, Xiaolong, 623, 1056  
Hu, Yue, 199  
Huang, Bo, 598, 719, 904, 1141  
Huang, Chu-Ren, 565  
Huu Pham, Quang, 888

Islam, Aadil, 667  
Ismail, Qusai, 661  
Iyer, Niveditha, 688, 1245

Jain, Aadish, 1276  
Jain, Abhinav, 875  
Jain, Sabhay, 1232

Jain, Vaibhav, 935  
Jarrar, Mustafa, 748  
Jiang, Lianxin, 623, 713, 1153, 1255  
Jiang, Yuxin, 793  
Jin, Meizhi, 1255  
Jindal, Aditya, 327  
Jindal, Vaibhav, 438  
Jung, Kyomin, 452  
Jurgens, David, 263  
  
Kaczyński, Konrad, 1027  
Kalach, Najla, 24  
Kanashiro Pereira, Lis, 17  
Kandasamy, Ilanthenral, 953, 1146  
Karasakalidis, Alexandros, 1125  
Karia, Neel, 387  
Karimi, Akbar, 220  
Kataria, Harsh, 941  
Kaushal, Ayush, 387  
Kaushik, Abhay, 211  
Kaushik, Vishesh, 327  
Kebriaei, Emad, 995  
Kestemont, Mike, 898  
Khan, Yakoob, 967  
Khloponin, Pavel PK, 404  
Khurana, Bholeshwar, 290  
Kilicoglu, Halil, 377  
Kim, Yanghoon, 452  
King, Milton, 650  
Klimczak, Hanna, 852  
Kobayashi, Ichiro, 17  
Kohler, Curt, 306  
Köksal, Abdullatif, 431  
Kolb, Peter, 632  
Kolis, Joanna, 974  
Kondrak, Grzegorz, 763  
Kotyushev, Mikhail, 913  
Kozlova, Olga, 927  
Krichene, Syrine, 423  
Kumar, Harshit, 438  
Kumar, Priyanshu, 743  
Kumar, Sandeep, 502  
Kumar, Sonal, 249  
Kurniawan, Kemal, 445  
Kuyumcu, Birol, 909  
  
Labadie, Roberto, 297  
Lack, Zander, 688  
Laparra, Egoitz, 348  
Lathiff, Nihatha, 404  
Laugier, Léo, 59  
Lee, Kathy J., 688  
  
Lee, Mark, 738  
Lefever, Els, 1051  
lei, yikun, 827  
Li, Chenliang, 1082  
Li, Junze, 275  
Li, Maochang, 623  
Li, Peiguang, 1032  
Li, Wei, 738  
Li, Wen, 1009  
Li, Wenjie, 565  
Li, Xiang, 521  
Li, Xuan, 1032  
Lian, Lianxin, 1056  
Liang, Huizhi, 799, 1179, 1303  
Liebeskind, Chaya, 138  
Liebeskind, Shmuel, 138  
Lin, Fangzhen, 793  
Lin, Jiaju, 1295  
Lin, Lei, 485  
Lin, Qihui, 521  
Lin, Xin, 183  
Lin, Zijie, 521  
Ling, Jing, 1295  
Ling, Zhenhua, 37  
Liu, Haoyang, 377  
Liu, Jiawei, 1295  
Liu, Jiaxiang, 99, 286  
Liu, Junfei, 258  
Liu, Kevin, 688  
Liu, Pai, 827  
Liu, Patrick, 688, 1245  
Liu, Pengyuan, 357  
Liu, Pingsheng, 183  
Liu, Quan, 37  
Liu, Renyuan, 281  
Liu, Tianshu, 870  
Liu, Xien, 416  
Liu, Yixiang, 357  
Liu, Zehao, 1179  
Logacheva, Varvara, 927  
López-Úbeda, Pilar, 984  
Lopez, Adam, 105  
Lu, Qin, 565  
Luo, Zhipeng, 578, 1130  
Luu, Son T., 846  
  
Ma, Jian, 713, 1153, 1255  
Ma, Weicheng, 667, 967  
Ma, Xinge, 478  
Magdy, Walid, 105  
Mahajan, Deepak, 511  
Mahajan, Diwakar, 317

Malik, Vijit, 327  
Mallick, Faraaz, 387  
Mallik, Arnob, 763  
Mamidi, Radhika, 1075, 1221  
Mangla, Karan, 1215  
Markchom, Thanet, 799, 1303  
Markov, Igor, 927  
Martelli, Federico, 24  
Martín-Valdivia, M. Teresa, 984  
Martin, Anna, 490  
Meaney, J. A., 105  
Menghwani, Preeti, 327  
Messina, Nicola, 1020  
Miller, Timothy, 348  
Miller, Tristan, 338  
Mishra, Vipul, 941  
Mittal, Abhishek, 175  
Mo, Yang, 623, 713, 1056, 1153, 1255  
Modi, Ashutosh, 175, 211, 290, 327, 438, 467, 511, 541, 1232  
Moens, Marie-Francine, 1015  
Mondal, Anik, 1169  
Montejo-Ráez, Arturo, 126  
Morozov, Dmitry, 913  
Mosquera, Alejandro, 554  
Movahedi, Sajad, 205  
Mukherjee, Ananta, 541  
Mukherjee, Sagnik, 541  
Müller, Thomas, 423  
Mundra, Jay, 511

Naghshnejad, Mina, 935  
Naim, Jannatun, 990, 1088  
Nakov, Preslav, 70  
Nanda, Ayush, 1185  
Nandy, Abhilash, 678  
Navigli, Roberto, 24  
Nguyen, Kiet, 919  
Nguyen, Ngan, 846  
Nguyen, Tam Minh, 888  
Nguyen, Thien Huu, 397  
Nguyen, Viet Anh, 888  
North, Kai, 548

Ororbia, Alexander, 833  
Ortega, Reynier, 297  
Ortiz-Zambrano, Jenny A., 126  
Osei-Brefo, Emmanuel, 1303  
Ouyang, Xuan, 286  
Özgür, Arzucan, 431

Paetzold, Gustavo Henrique, 1, 617

Pal, Avik, 290  
Palliser-Sans, Rafel, 960  
Palomino, Marco, 860  
Pan, Chunguang, 578, 1130  
Panchenko, Alexander, 157, 927  
Pandey, Chandan Kumar, 1215  
Pandey, Harshit, 189, 233  
Pang, Chao, 286  
Paraschiv, Andrei, 225  
Patwal, Suraj, 502  
Pavlopoulos, John, 59  
Pedersen, Ted, 364, 490  
Peng, Wei, 199  
Piersa, Jarosław, 974  
Pintilie, Cosmin, 787  
Plank, Barbara, 338  
Plaza-del-Arco, Flor Miriam, 984  
Pluciński, Kamil, 852  
Poesio, Massimo, 338  
Pokala, Sai Mahesh, 678  
Ponomareva, Maria, 163  
Pouran Ben Veyseh, Amir, 397  
Prati, Andrea, 220  
Pritzkau, Albert, 1037  
Przybyła, Piotr, 1027  
Pu, Pearl, 275

Qarqaz, Ahmed, 1068  
Qi, Zhiyuan, 1239  
Quang Dao, Huy, 888

Rachinskiy, Maxim, 756  
Raha, Tathagata, 1221  
Rahgooy, Taher, 948  
Rahgouy, Mostafa, 948  
Rajendram, S Milton, 1185  
Ranasinghe, Tharindu, 771, 833  
Ranjbar, Niloofar, 724  
Rao, Gang, 623, 1056  
Ratadiya, Pratik, 1276  
Rathi, Abhishek, 1276  
Razmara, Jafar, 948  
Razzhigaev, Anton, 157  
Ren, Junsong, 1056  
Rey, Alexey, 780  
Rial-Farràs, Albert, 960  
Rivas Rojas, Kervy, 144  
Rodriguez, Mariano Jason, 297  
Roele, Cees, 270  
Rosenthal, Sara, 317  
Rossi, Leonardo, 220  
Rosso, Paolo, 297

Rotaru, Armand, 655  
Rozi, Erik, 688, 1245  
Ruan, Xiaoyi, 1153, 1255  
Ruan, Yu-Ping, 37  
Ruan, Zhihao, 1056  
Rusert, Jonathan, 881  
Russo, Irene, 694

S, Angel Deborah, 1185  
Sabri, Nazanin, 995  
Salemi, Alireza, 995  
Samoray, Nicholas, 1190  
Samson, Mihai, 1226  
Sarkar, Diptanu, 833  
Sarol, M. Janina, 377  
Sarthak, Sarthak, 805  
Satława, Michał, 974  
Scerri, Antony, 306  
Schulz, Philip, 445  
Semenov, Nikita, 927  
Shaar, Shaden, 70  
Shah, Jinang, 438  
Shailabh, Shashank, 467  
Shakery, Azadeh, 205, 995  
Shan, Lili, 485  
Shandhilya, Tushar, 541  
Shardlow, Matthew, 1, 603  
Sharma, Abheesht, 189, 233  
Sharma, Mayukh, 953, 1146  
Sharma, Raksha, 1061, 1169  
Shen, Jianping, 1056  
Shen, Jianping, 623, 713  
Shirude, Neil, 541  
Shou, Ziyi, 793  
Shrivastava, Manish, 1262  
Shukla, Shikhar, 805  
Sileo, Damien, 1015  
Silvestri, Fabrizio, 70  
Simpson, Edwin, 338  
Singh, Aadarsh, 743  
Singh, Abrit Pal, 1185  
Singh, Chirag, 1215  
Singh, Pranaydeep, 1051  
Sivanaiah, Rajalakshmi, 1185  
Smădu, Răzvan-Alexandru, 1160  
Smolenska, Greta, 632  
Song, Bingyan, 578, 1130  
Sorensen, Jeffrey, 59  
Sorgente, Antonio, 560  
Sourav, Shubham, 1232  
Srivastava, Jaya, 327  
Stodden, Regina, 640

Strohmaier, David, 590, 730  
Stymne, Sara, 150  
Su, Weiyue, 286  
Su, Xin, 348, 458  
Su, Yinpei, 51  
Su, Yu, 485  
Sultana, Afrin, 1204  
Suman, Thakur Ashutosh, 875  
Sun, Chengjie, 485  
Sun, Jingyi, 521  
Sun, Jinquan, 1283  
Sun, Xian, 1032  
Sun, Yu, 99, 286  
Suthaharan, Shan, 233

Tang, Jiji, 99  
Tang, Jillian, 688  
Tang, Sinan, 632  
Tao, Xin, 1175  
Tapuc, Andrada, 787  
Tasneem, Fareen, 990, 1088  
Tasnia, Radiathun, 990, 1088  
Taya, Yuki, 17  
Tayyar Madabushi, Harish, 243, 738  
Thanh Nguyen, Luan, 919  
Therien, Benjamin, 410  
Tian, Junfeng, 1082  
Tissot, Hegler, 683  
Tola, Gabriele, 24  
Trandabat, Diana, 787  
TT, Mirnalinee, 1185  
Tyagi, Lakshay, 290  
Tyen, Gladys, 590

Uma, Alexandra, 338  
Upadhyay, Ishan Sanjeev, 1221  
Ureña-López, L. Alfonso, 984  
Uzuner, Özlem, 348

Valencia-García, Rafael, 1096  
Varma, Harshit, 1276  
Varma, Vasudeva, 1221  
Vasantha, W.B., 953, 1146  
Venugopal, Gayatri, 640  
Vettigli, Giuseppe, 560  
Voskoboinik, Katja, 700  
Vosoughi, Soroush, 667, 967

Wang, Chenyi, 870  
Wang, Jin, 478, 841, 1045, 1289  
Wang, Linlin, 183  
Wang, Nancy X. R., 317  
Wang, Qijun, 793

Wang, Shaojun, 820  
Wang, Shengguang, 578, 1130  
Wang, Shuohuan, 286  
Wang, Weikang, 357  
Wang, Yanmeng, 820  
Wang, Ye, 820  
Wang, Yong, 810  
Wang, Yu, 1283  
Wang, Zhen, 258  
Wang, Zhiwei, 1295  
Wei, Si, 37  
Wilson, Steven, 105  
Wu, Hao, 793  
Wu, Ji, 416  
Wu, Yi, 357  
  
Xiang, Rong, 565  
Xiang, Yuejia, 1239  
Xiao, Jing, 820  
Xiao, Wenming, 1082  
Xie, Shuyi, 713, 1153  
Xie, Wanying, 706  
Xie, Xin, 810  
Xie, Yubo, 275  
Xie, Yuqiang, 199  
Xing, Luxi, 199  
Xu, Ruifeng, 521  
  
Yan, Erik, 243  
Yan, Ming, 1082  
Yang, Haiqin, 713, 1153, 1255  
Yang, Maoqin, 1120  
Yang, Xiaoyu, 37  
Yıldırım, Bekir, 431  
Yin, Weichong, 99  
Yoon, Sangwon, 452  
You, Huiling, 150  
Yu, Zhewen, 1289  
Yuan, Zheng, 590, 730  
Yüksel, Yusuf, 431  
  
Zaharia, George-Eduard, 534, 609  
Zamłyńska, Katarzyna, 974  
Zampieri, Marcos, 1, 548, 833  
Zeinali, Hossein, 724  
Zeng, Bo, 820  
Zhang, Genyu, 485  
Zhang, Jing, 51  
Zhang, Lei, 1283  
Zhang, Ningyu, 810  
Zhang, Qi, 1283  
Zhang, Xuejie, 478, 841, 1045, 1289

Zhang, Yice, 521  
Zhang, Yunyan, 1239  
Zhao, Qian, 183  
Zhao, Tiejun, 870  
Zhao, Yingjia, 1175  
Zhao, Yiyun, 348, 458  
Zheng, Boyuan, 37  
Zheng, Yefeng, 1239  
Zhestiankin, Boris, 163  
Zhou, Kaiyin, 416  
Zhou, Mengyuan, 1153, 1255  
Zhou, Xiaobing, 281, 598, 719, 904  
Zhou, Xiaobing ZXB, 1108  
Zhou, Yuxuan, 416  
Zhu, Haijun, 820  
Zhu, Qinglin, 521  
Zhu, Xiaodan, 37, 416  
Zhu, Xingran, 150  
Zhu, Xingyu, 1045  
Zhuang, Yimeng, 51  
Zou, Liang, 1009  
Zylich, Brian, 1190