

EMNLP 2022

EMNLP 2022 Industry Track

Proceedings of the Conference

December 7 - 11, 2022

©2022 The Association for Computational Linguistics

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

ISBN 978-1-952148-25-5

Organizing Committee

Angeliki Lazaridou, DeepMind
Yun Yao Li, Apple

Program Committee

Area Chairs

Alan Akbik, Humboldt-Universität zu Berlin
Jennifer Chu-Carroll, Elemental Cognition
Lisa Anne Hendricks, DeepMind
Marina Danilevsky, IBM Research
Mo Yu, IBM Research
Rishita Anubhai, Amazon
Ruoming Pang, Apple
Sachin Agarwal, Apple
Shimei Pan, UMBC
Srinivas Bangalore, Interactions Corp
Xiaoqiang Luo, Google

Program Committee

Abdalghani Abujabal, Amazon Alexa AI
Abeba Birhane, University College Dublin
Abhishek Singh, Samsung
Ai Ti Aw, Institute for Infocomm Research
Alex Marin, Microsoft Corporation
Alicia Sagae, Research Scientist
Amar Prakash Azad, IBM AI Research
Anastassia Loukina, Grammarly Inc
Anil Ramakrishna, Amazon
Anjishnu Kumar, Amazon Alexa
Ankur Gandhe, Amazon

Ankush Gupta, IBM Research
Anmol Goel, IIIT Hyderabad
Anna Lisa Gentile, IBM Research Almaden
Anusha Balakrishnan, Inflection AI
Aoife Cahill, Dataminr
Aparna Elangovan, The University of Melbourne
Arun Babu, Facebook
Ashish Shenoy, Meta
Aswarth Abhilash Dara, Amazon
Avneesh Saluja, Netflix
Benjamin Rozonoyer, University of Massachusetts Amherst
Bowei Zou, Institute for Infocomm Research
Brian Ulicny, Raytheon BBN Technologies
Budhaditya Deb, Microsoft Corporation
Byung-Hak Kim, AKASA
Chanjun Park, Upstage
Chen Liu, Technische Universität Darmstadt
Cheoneum Park, Hyundai Motor Group
Claudia Borg, University of Malta
Dakuo Wang, IBM Research
Damianos Karakos, Raytheon BBN Technologies
David Melville, Elemental Cognition
David Elson, Google
David Uthus, Google Research
Daxin Jiang, STCA, Microsoft
Deborah Dahl, Conversational Technologies
Deepak Muralidharan, Apple
Dejing Dou, University of Oregon
Derrick Higgins, Illinois Institute of Technology
Deyi Xiong, Tianjin University
Dongchan Kim, Amazon Alexa
Dookun Park, Amazon Alexa
Ehud Reiter, University of Aberdeen
Elio Querze, Bose Corp
Emre Barut, Alexa AI
Enrique Henestroza Anguiano, Ask Media Group
Eshwar Shamanna Girishekar, Amazon
Estevam Hruschka, Megagon Labs - <https://megagon.ai/>
Ethan Selfridge, LivePerson
Eunah Cho, Amazon, Alexa AI
Fabio Mercorio, University of Milano-Bicocca
Faisal Ladhak, Columbia University
Feifei Pan, Rensselaer Polytechnic Institute
Frank Schilder, Thomson Reuters
Frederic Mailhot, Dialpad, Inc
Geewook Kim, NAVER
Guoyin Wang, Amazon Alexa AI
Han Li, Amazon
Hassan Sawaf, aixplain, inc.
He Xie, Amazon Alexa AI
Hemant Misra, Swiggy (BundlTechnologies)

Hidetaka Kamigaito, Nara Institute of Science and Technology
Honglei Guo, Tsinghua University
ilknur Durgar Elkahlout, TURKCELL
Isabel Trancoso, INESC-ID / IST Univ. Lisbon
Ishan Jindal, IBM Research
Ismini Lourentzou, Virginia Tech
Jacopo Tagliabue, Tooso
Jade Abbott, Retro Rabbit
Jaegul Choo, KAIST
Jaesik Choi, Korea Advanced Institute of Science and Technology
Jaime Lorenzo-Trueba, Amazon
Jared Kramer, Amazon
Jesse Vig, Salesforce Research
Jiangning Chen, Amazon
Jiaze Chen, Bytedance AI Lab
Jie Ma, AWS AI Lab
John Chen, Interactions LLC
Judith Gaspers, Amazon
Jun Seok Kang, Blink Health
Justin Chiu, Rakuten USA
Kai Yu, Shanghai Jiao Tong University
Kalpa Gunaratna, Samsung Research America
Kartik Mehta, Amazon
Kasturi Bhattacharjee, AWS AI, Amazon
Kazuya Kawakami, University of Oxford
Keith Trnka, 98point6
Kfir Bar, College of Management Academic Studies
Khadige Abboud, Amazon Alexa AI
Kunho Kim, Microsoft Corporation
Laurel Orr, Stanford
Lee Becker, Educational Testing Service
Lei Chen, Rakuten
Lei Shu, Google Research
Li Dong, Amazon.com
Liangming Pan, National University of Singapore
Lin Pan, Amazon
Ling Tsou, Zoom Video Communications
Lingjia Deng, Bloomberg L.P.
Lisheng Fu, Amazon
Long Qin, Alibaba
Lorenzo Malandri, University of Milan - Bicocca
Luoxin Chen, Amazon Alexa AI
Lynette Hirschman, MITRE
Maeda Hanafi, IBM
Mahnoosh Mehrabani, Interactions LLC
Marek Suppa, Comenius University in Bratislava
Margot Mieskes, University of Applied Sciences, Darmstadt
MARK SAMMONS, Elemental Cognition
Matthew Mulholland, Educational Testing Service
Michael Flor, Educational Testing Service
Michal Shmueli-Scheuer, IBM Research

Mingyue Shang, Amazon
Mohamed Abdelhady, Amazon
Mohamed AlTantawy, Agolo
Mohammad Kachuee, Amazon Alexa AI
Monica Sunkara, Amazon
Narges Tabari, AWS AI Labs, Amazon
Navid Nobani, University of Milano-Bicocca
Ngoc Phuoc An Vo, IBM Research
Nikhil Rasiwasia, Amazon.com
Nikita Bhutani, Megagon Labs
Nitin Madnani, Educational Testing Service
Nyalleng Moorosi, Google AI
Pablo Duboue, Textualization Software Ltd.
Pengfei Liu, Centre for Perceptual and Interactive Intelligence
Pengfei Li, Nanyang Technological University
Peter Grasch, Apple Inc.
Petr Lorenc, Czech Technical University in Prague
Poornima Chozhiyath Raman, Roku
Prasanna kumar Muthukumar, BBN Technologies
Qingkai Zeng, University of Notre Dame
Radhika Gaonkar, Microsoft Search, Assistant and Intelligence
Radityo Eko Prasajo, Pitik.id
Rahul Divekar, Educational Testing Service
Ramy Eskander, Twitter
Rutuja Ubale, Educational Testing Service
Sagnik Ray Choudhury, University of Michigan
Saleh Soltan, Amazon Alexa
Saloni Potdar, IBM Watson, Carnegie Mellon University
Sandesh Swamy, Amazon
Sangameshwar Patil, TRDDC, TCS Research and Innovation
Sanjeev Kumar, Quark.ai
Sanjika Hewavitharana, eBay
Sarah Campbell, Amazon Alexa AI
Sarasi Lalithsena, IBM
Sashank Santhanam, University of North Carolina at Charlotte/ Apple
Saurabh Khanwalkar, Course Hero
Sergio Oramas, Pandora
Shashank Gupta, Microsoft Search, Assistant and Intelligence
Shuangyong Song, JD AI Research
Shuyan Dong, Meta
Siddharth Varia, Amazon
Siddharth Patwardhan, Apple
Sidharth Mudgal, Google, Inc.
Sopan Khosla, Amazon Web Services, Amazon Inc
Sourish Chaudhuri, Google Inc
Spyros Matsoukas, Amazon.com
Stavroula Skylaki, Thomson Reuters Labs
Stephen Pulman, Apple Inc.
Stevie Bergman, DeepMind
Sudarshan R. Thitte, IBM
Sumanth Prabhu, Applied Research Department, Swiggy (BUNDL Technologies), Bangalore, Karnataka,

India - 560103
Sun Kim, Naver; NCBI/NIH
Sunayana Sitaram, Microsoft Research India
Tao Wang, ByteDance AI Lab
Tara Taghavi, Amazon
Tarec Fares, Bloomberg LP
Tilman Becker, DFKI
Tirthankar Dasgupta, Tata Consultancy Services Ltd.
Tong Guo, Meituan
Tong Wang, Amazon
varun embar, Apple Inc.
Varun Kumar, Amazon Alexa
Varun Nagaraj Rao, Princeton University
Victor Soto, Amazon Inc.
Vinayshekhar Bannihatti Kumar, AWS AI
Vitobha Munigala, Research Engineer, IBM Research
Wei Zhu, ECNU
Weiwei Guo, LinkedIn
Wonseok Hwang, LBox
Xian-Ling Mao, Beijing Institute of Technology
Xiao Yang, Meta Platforms
Xiaohu Liu, Amazon
Xiliang Zhu, Dialpad
Xuan Zhu, Amazon
Xuye Liu, University of Waterloo
Yi-Chia Wang, Facebook AI
Yichao Zhou, Google Research
Yinfei Yang, Google
Yoav Katz, IBM Research AI
Youngja Park, IBM T. J. Watson Research Center
Yu Wang, Samsung Research America
Yuanfeng Song, Hong Kong University of Science and Technology, WeBank Co., Ltd
Yuji Matsumoto, Riken Center for Advanced Intelligence Project
Yuval Marton, University of Washington
Ziming Huang, Tencent

Table of Contents

<i>Unsupervised Term Extraction for Highly Technical Domains</i> Francesco Fusco, Peter Staar and Diego Antognini	1
<i>DynaMaR: Dynamic Prompt with Mask Token Representation</i> Xiaodi Sun, Sunny Rajagopalan, Priyanka Nigam, Weiyi Lu, Yi Xu, Iman Keivanloo, Belinda Zeng and Trishul Chilimbi	9
<i>A Hybrid Approach to Cross-lingual Product Review Summarization</i> Saleh Soltan, Victor Soto, Ke Tran and Wael Hamza	18
<i>Augmenting Operations Research with Auto-Formulation of Optimization Models From Problem Descriptions</i> Rindra Ramamonjison, Haley Li, Timothy Yu, Shiqi HE, Vishnu Rengan, Amin Banitalebi-Dehkordi, Zirui Zhou and Yong Zhang	29
<i>Knowledge Distillation based Contextual Relevance Matching for E-commerce Product Search</i> Ziyang Liu, Chaokun Wang, Hao Feng, Lingfei Wu and Liqun Yang	63
<i>Accelerating the Discovery of Semantic Associations from Medical Literature: Mining Relations Between Diseases and Symptoms</i> Alberto Purpura, Francesca Bonin and Joao Bettencourt-Silva	77
<i>PENTATRON: PErsonalized coNText-Aware Transformer for Retrieval-based cOnversational uNderstanding</i> Niranjan Uma Naresh, Ziyang Jiang, Ankit Ankit, Sungjin Lee, Jie Hao, Xing Fan and Chenlei Guo	90
<i>Machine translation impact in E-commerce multilingual search</i> Bryan Zhang and amita misra	99
<i>Ask-and-Verify: Span Candidate Generation and Verification for Attribute Value Extraction</i> Yifan Ding, Yan Liang, Nasser Zalmout, Xian Li, Christan Grant and Tim Wenginger	110
<i>Consultation Checklists: Standardising the Human Evaluation of Medical Note Generation</i> Aleksandar Savkov, Francesco Moramarco, Alex Papadopoulos Korfiatis, Mark Perera, Anya Belz and Ehud Reiter	121
<i>Towards Need-Based Spoken Language Understanding Model Updates: What Have We Learned?</i> Quynh Do, Judith Gaspers, Daniil Sorokin and Patrick Lehnen	131
<i>Knowledge Distillation Transfer Sets and their Impact on Downstream NLU Tasks</i> Charith Peris, Lizhen Tan, Thomas Gueudre, Turan Gojayev, Pan Wei and Gokmen Oz	138
<i>Exploiting In-Domain Bilingual Corpora for Zero-Shot Transfer Learning in NLU of Intra-Sentential Code-Switching Chatbot Interactions</i> Maia Aguirre, Manex Serras, Laura García-Sardiña, Jacobo López-Fernández, Ariane Méndez and Arantza del Pozo	148
<i>Calibrating Imbalanced Classifiers with Focal Loss: An Empirical Study</i> Cheng Wang, Jorge Balazs, György Szarvas, Patrick Ernst, Lahari Poddar and Pavel Danchenko	155

<i>Unsupervised training data re-weighting for natural language understanding with local distribution approximation</i>	
Jose Garrido Ramas, Dieu-Thu Le, Bei Chen, Manoj Kumar and Kay Rottmann	164
<i>Cross-Encoder Data Annotation for Bi-Encoder Based Product Matching</i>	
Justin Chiu and Keiji Shinzato	171
<i>Deploying a Retrieval based Response Model for Task Oriented Dialogues</i>	
Lahari Poddar, György Szarvas, Cheng Wang, Jorge Balazs, Pavel Danchenko and Patrick Ernst	179
<i>Tackling Temporal Questions in Natural Language Interface to Databases</i>	
Ngoc Phuoc An Vo, Octavian Popescu, Irene Manotas and Vadim Sheinin	189
<i>Multi-Tenant Optimization For Few-Shot Task-Oriented FAQ Retrieval</i>	
Asha Vishwanathan, Rajeev Warriar, Gautham Vadakkekara Suresh and Chandra Shekhar Kandpal	198
<i>Iterative Stratified Testing and Measurement for Automated Model Updates</i>	
Elizabeth Dekeyser, Nicholas Comment, Shermin Pei, Rajat Kumar, Shruti Rai, Fengtao Wu, Lisa Haverty and Kanna Shimizu	208
<i>SLATE: A Sequence Labeling Approach for Task Extraction from Free-form Inked Content</i>	
Apurva Gandhi, Ryan Serrao, Biyi Fang, Gilbert Antonius, Jenna Hong, Tra My Nguyen, Sheng Yi, Ehi Nosakhare, Irene Shaffer, Soundararajan Srinivasan and Vivek Gupta	216
<i>Gaining Insights into Unrecognized User Utterances in Task-Oriented Dialog Systems</i>	
Ella Rabinovich, Matan Vetzler, David Boaz, Vineet Kumar, Gaurav Pandey and Ateret Anaby Tavor	228
<i>CoCoID: Learning Contrastive Representations and Compact Clusters for Semi-Supervised Intent Discovery</i>	
Qian Cao, Deyi Xiong, Qinlong Wang and Xia Peng	236
<i>Tractable & Coherent Multi-Document Summarization: Discrete Optimization of Multiple Neural Modeling Streams via Integer Linear Programming</i>	
Litton J Kurisinkel and Nancy Chen	247
<i>Grafting Pre-trained Models for Multimodal Headline Generation</i>	
Lingfeng Qiao, Chen Wu, Ye Liu, haoyuan peng, di yin and Bo Ren	254
<i>Semi-supervised Adversarial Text Generation based on Seq2Seq models</i>	
Hieu Le, Dieu-Thu Le, Verena Weber, Chris Church, Kay Rottmann, Melanie Bradford and Peter Chin	264
<i>Is it out yet? Automatic Future Product Releases Extraction from Web Data</i>	
Gilad Fuchs, Ido Ben-Shaul and Matan Mandelbrod	273
<i>Automatic Scene-based Topic Channel Construction System for E-Commerce</i>	
Peng Lin, Yanyan Zou, Lingfei Wu, Mian Ma, Zhuoye Ding and Bo Long	282
<i>SpeechNet: Weakly Supervised, End-to-End Speech Recognition at Industrial Scale</i>	
Raphael Tang, Karun Kumar, Gefei Yang, Akshat Pandey, Yajie Mao, Vladislav Belyaev, Madhuri Emmadi, Craig Murray, Ferhan Ture and Jimmy Lin	295

<i>Controlled Language Generation for Language Learning Items</i> Kevin Stowe, Debanjan Ghosh and Mengxuan Zhao	304
<i>Improving Text-to-SQL Semantic Parsing with Fine-grained Query Understanding</i> Jun Wang, Patrick Ng, Alexander Hanbo Li, Jiarong Jiang, Zhiguo Wang, Bing Xiang, Ramesh Nallapati and Sudipta Sengupta	316
<i>Unsupervised Dense Retrieval for Scientific Articles</i> Dan Li, Vikrant Yadav, Zubair Afzal and George Tsatsaronis	323
<i>Learning Geolocations for Cold-Start and Hard-to-Resolve Addresses via Deep Metric Learning</i> Govind . and Saurabh Sohoney	332
<i>Meta-learning Pathologies from Radiology Reports using Variance Aware Prototypical Networks</i> Arijit Sehanobish, Kawshik Kannan, Nabila Abraham, Anasuya Das and Benjamin Odry	342
<i>Named Entity Recognition in Industrial Tables using Tabular Language Models</i> Aneta Koleva, Martin Ringsquandl, Mark Buckley, Rakeb Hasan and Volker Tresp	358
<i>Reinforced Question Rewriting for Conversational Question Answering</i> Zhiyu Chen, Jie Zhao, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko and Shervin Malmasi	367
<i>Improving Large-Scale Conversational Assistants using Model Interpretation based Training Sample Selection</i> Stefan Schroedl, Manoj Kumar, Kiana Hajebi, Morteza Ziyadi, Sriram Venkatapathy, Anil Ramakrishna, Rahul Gupta and Pradeep Natarajan	381
<i>Improving Precancerous Case Characterization via Transformer-based Ensemble Learning</i> Yizhen Zhong, Jiajie Xiao, Thomas Vetterli, Mahan Matin, Ellen Loo, Jimmy Lin, Richard Bourgon and Ofer Shapira	389
<i>Developing Prefix-Tuning Models for Hierarchical Text Classification</i> Lei Chen, Houwei Chou and Xiaodan Zhu	400
<i>PAIGE: Personalized Adaptive Interactions Graph Encoder for Query Rewriting in Dialogue Systems</i> Daniel Biś, Saurabh Gupta, Jie Hao, Xing Fan and Chenlei Guo	408
<i>Fast Vocabulary Transfer for Language Model Compression</i> Leonidas Gee, Andrea Zugarini, Leonardo Rigutini and Paolo Torrioni	419
<i>Multimodal Context Carryover</i> Prashan Wanigasekara, Nalin Gupta, Fan Yang, Emre Barut, Zeynab Raeesy, Kechen Qin, Stephen Rawls, Xinyue Liu, Chengwei Su and Spurthi Sandiri	427
<i>Distilling Multilingual Transformers into CNNs for Scalable Intent Classification</i> Besnik Fetahu, Akash Veeragouni, Oleg Rokhlenko and Shervin Malmasi	439
<i>Bringing the State-of-the-Art to Customers: A Neural Agent Assistant Framework for Customer Service Support</i> Stephen Obadinma, Faiza Khan Khattak, Shirley Wang, Tania Sidhorn, Elaine Lau, Sean Robertson, Jingcheng Niu, Winnie Au, Alif Munim, Karthik Raja Kalaiselvi Bhaskar, Bencheng Wei, Iris Ren, Muhammad Waqar, Erin Li, Bukola Ishola, Michael Wang, Griffin Tanner, Yu-Jia Shiah, Sean X. Zhang, Kwesi Apponsah, Kanishk Patel, Jaswinder Narain, Pandya Deval, Xiaodan Zhu, Frank Rudzicz and Elham Dolatabadi	450

<i>Zero-Shot Dynamic Quantization for Transformer Inference</i> Yousef El-Kurdi, Jerry Quinn and Avi Sil	461
<i>Fact Checking Machine Generated Text with Dependency Trees</i> Alex Estes, Nikhita Vedula, Marcus Collins, Matt Cecil and Oleg Rokhlenko	468
<i>Prototype-Representations for Training Data Filtering in Weakly-Supervised Information Extraction</i> Nasser Zalmout and Xian Li	477
<i>CGF: Constrained Generation Framework for Query Rewriting in Conversational AI</i> Jie Hao, Yang Liu, Xing Fan, Saurabh Gupta, Saleh Soltan, Rakesh Chada, Pradeep Natarajan, Chenlei Guo and Gokhan Tur	485
<i>Entity-level Sentiment Analysis in Contact Center Telephone Conversations</i> Xue-Yong Fu, Cheng Chen, Md Tahmid Rahman Laskar, Shayna Gardiner, Pooja Hiranandani and Shashi Bhushan TN	494
<i>QUILL: Query Intent with Large Language Models using Retrieval Augmentation and Multi-stage Distillation</i> Krishna Srinivasan, Karthik Raman, Anupam Samanta, Lingrui Liao, Luca Bertelli and Michael Bendersky	502
<i>Distinguish Sense from Nonsense: Out-of-Scope Detection for Virtual Assistants</i> Cheng Qian, Haode Qi, Gengyu Wang, Ladislav Kunc and Saloni Potdar	512
<i>PLATO-Ad: A Unified Advertisement Text Generation Framework with Multi-Task Prompt Learning</i> Zeyang Lei, Chao Zhang, Xinchao Xu, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, Yi Yang and Shuanglong Li	522
<i>Dense Feature Memory Augmented Transformers for COVID-19 Vaccination Search Classification</i> Jai Gupta, Yi Tay, Chaitanya Kamath, Vinh Tran, Donald Metzler, Shailesh Bavadekar, Mimi Sun and Evgeniy Gabrilovich	531
<i>Full-Stack Information Extraction System for Cybersecurity Intelligence</i> Youngja Park and Taesung Lee	541
<i>Deploying Unified BERT Moderation Model for E-Commerce Reviews</i> Ravindra Nayak and Nikesh Garera	550
<i>SimANS: Simple Ambiguous Negatives Sampling for Dense Text Retrieval</i> Kun Zhou, Yeyun Gong, Xiao Liu, Wayne Xin Zhao, Yelong Shen, Anlei Dong, Jingwen Lu, Rangan Majumder, Ji-Rong Wen, Nan Duan and Weizhu Chen	558
<i>Revisiting and Advancing Chinese Natural Language Understanding with Accelerated Heterogeneous Knowledge Pre-training</i> Taolin Zhang, junwei dong, Jianing Wang, Chengyu Wang, Ang Wang, Yinghui Liu, jun huang, Yong Li and XIAOFENG HE	570
<i>A Stacking-based Efficient Method for Toxic Language Detection on Live Streaming Chat</i> Yuto Oikawa, Yuki Nakayama and Koji Murakami	581
<i>End-to-End Speech to Intent Prediction to improve E-commerce Customer Support Voicebot in Hindi and English</i> Abhinav Goyal, Anupam Singh and Nikesh Garera	589

<i>PILE: Pairwise Iterative Logits Ensemble for Multi-Teacher Labeled Distillation</i>	
Lianshang Cai, Linhao Zhang, Dehong Ma, Jun Fan, Daiting Shi, Yi Wu, Zhicong Cheng, Simiu Gu and Dawei Yin	597
<i>A Comprehensive Evaluation of Biomedical Entity-centric Search</i>	
Elena Tutubalina, Zulfat Miftahutdinov, Vladimir Muravlev and Anastasia Shneyderman	606
<i>Domain Adaptation of Machine Translation with Crowdworkers</i>	
Makoto Morishita, Jun Suzuki and Masaaki Nagata	616
<i>Biomedical NER for the Enterprise with Distillated BERN2 and the Kazu Framework</i>	
Wonjin Yoon, Richard Jackson, Elliot Ford, Vladimir Poroshin and Jaewoo Kang	629
<i>Large-scale Machine Translation for Indian Languages in E-commerce under Low Resource Constraints</i>	
Amey Patil and Nikesh Garera	637
<i>Topic Modeling by Clustering Language Model Embeddings: Human Validation on an Industry Dataset</i>	
Anton Eklund and Mona Forsman	645

Unsupervised Term Extraction for Highly Technical Domains

Francesco Fusco
IBM Research
ffu@zurich.ibm.com

Peter Staar
IBM Research
taa@zurich.ibm.com

Diego Antognini
IBM Research
Diego.Antognini@ibm.com

Abstract

Term extraction is an information extraction task at the root of knowledge discovery platforms. Developing term extractors that are able to generalize across very diverse and potentially highly technical domains is challenging, as annotations for domains requiring in-depth expertise are scarce and expensive to obtain. In this paper, we describe the term extraction subsystem of a commercial knowledge discovery platform that targets highly technical fields such as pharma, medical, and material science. To be able to generalize across domains, we introduce a *fully unsupervised* annotator (UA). It extracts terms by combining novel morphological signals from sub-word tokenization with term-to-topic and intra-term similarity metrics, computed using general-domain pre-trained sentence-encoders. The annotator is used to implement a *weakly-supervised setup*, where transformer-models are fine-tuned (or pre-trained) over the training data *generated* by running the UA over large unlabeled corpora. Our experiments demonstrate that our setup can improve the predictive performance *while decreasing* the inference latency on both CPUs and GPUs. Our annotators provide a very competitive baseline for all the cases where annotations are not available.

1 Introduction

Automated Term Extraction (ATE) is the task of extracting terminology from domain-specific corpora. Term extraction is the most important information extraction task for knowledge discovery systems – whose aim is to create structured knowledge from unstructured text – because domain specific terms are the linguistic representation of domain-specific concepts. To be of use in knowledge discovery systems (e.g., SAGA (Ilyas et al., 2022), DeepSearch (Dognin et al., 2020)) the term extraction has to identify individual *mentions* of terms to enable downstream components (i.e., the entity

Wikipedia

Text from <https://en.wikipedia.org/wiki/JPEG>.

JPEG (ⁱ²/^{dʒɛɪɪpɛɡ}/^{jaɪ-pɛɡ}²)² is a commonly used method of [lossy compression](#) for [digital images](#), particularly for those images produced by [digital photography](#).

Our unsupervised term-extractor annotator

TEXT = JPEG (ⁱ^{dʒɛɪɪpɛɡ}/^{jaɪ-pɛɡ}²)² is a commonly used Method of lossy compression for digital images, particularly for those images produced by digital photography.

[JPEG]	START=0	END=4	Confidence=0.60
[JAY-peg]	START=17	END=24	Confidence=0.90
[lossy compression]	START=58	END=75	Confidence=0.73
[digital images]	START=80	END=94	Confidence=0.93
[digital photography]	START=138	END=157	Confidence=0.92

Figure 1: Our term extractor identifies the same mentions as Wikipedia without *relying on annotated data*.

linker) to use not only the terms, but also their surrounding context. Unlike other applications of term extraction, such as text classification, where it is sufficient to extract representative terms for entire documents or even use generative approaches, term extraction in knowledge discovery systems has to be approached as a sequence tagging task.

The largest challenges for term extraction systems, when used for knowledge discovery, are generalization across domains and lack of annotated data. In fact, commercial knowledge discovery platforms are typically required to process large corpora targeting very diverse and often highly technical domains. Organizing annotation campaigns for such vertical domains is a costly process as it requires highly specialized domain experts. An additional challenge for such platforms are the computational requirements, which must be accounted for when developing technologies required to sift through very large and often proprietary corpora.

In this work, we describe an effective term extraction approach used in a commercial knowledge discovery platform¹ to extract *Wikipedia-like concepts*² from text (see Figure 1). Our approach does

¹<https://ds4sd.github.io>.

²The linking from words to Wikilinks is done manually on Wikipedia, see https://en.wikipedia.org/wiki/Wikipedia:Manual_of_Style/Linking for more details.

not require any human annotation, offers the flexibility to select the right trade-off between accuracy and inference latency, and enables the deployment of lightweight models running entirely on CPUs.

At its core, our approach is a *weakly supervised* setup (see Figure 2), where transformer models are fine-tuned (or even entirely pre-trained) using the *weak labels* generated by a *fully unsupervised* term annotator. The unsupervised annotator (UA) combines novel morphological and semantic signals to tag sequences of text corresponding to domain-specific terminology. In fact, in addition to part-of-speech tagging to identify candidate terms, the UA exploits sub-word tokenization techniques – commonly used in language models to highlight words that are outside of the common vocabulary – to indirectly measure the morphological complexity of a word based on its sub-tokens. To the best of our knowledge, this is the first work relying on sub-word tokenization units in the context of term extraction. To prune the candidate set of terms the annotator uses two semantic metrics as thresholds: the *topic-score* and a novel *specificity score* that are both computed using representations from sentence encoders. The unsupervised annotator, combined with the two-stage weakly supervised setup, makes our approach particularly attractive for practical industrial setups because computationally intensive techniques used by the unsupervised annotator are not paid at inference time. Therefore, one can improve the annotation quality by using more expensive techniques (e.g., entity linking to external knowledge bases), without adding costs at inference time. The two main contributions of this paper are summarized as follows:

1. We extract a novel morphology signal from subword-unit tokenization and we introduce a new metric called the *specificity score*. Upon those signals, we build an unsupervised term-extractor that offers competitive results when no annotation is available.
2. We show that by fine-tuning transformer models over the weak labels produced by the unsupervised term extractor we decrease the latency and improve the prediction quality.

2 Related work

Automated Term Extraction (ATE) is a natural language processing task that has been the subject of many research studies (Buitelaar et al., 2005;

Lossio-Ventura et al., 2016; Zhang et al., 2018; Ma et al., 2019; Šajatović et al., 2019). What we describe in this work is an effective term extraction approach that is fully unsupervised and also offers the flexibility and modularity to deploy and easily maintain systems in production.

ATE should not be confused with keyphrase extraction (Firoozeh et al., 2020; Mahata et al., 2018; Bennani-Smires et al., 2018) and keyphrase generation (Wu et al., 2022; Chen et al., 2020), which have the goal of extracting, or generating, key phrases that best describe a given free text document. Keyphrases can be seen as a set of tags associated to a document. In the context of keyphrase extraction, sentence embedders have been used in the literature, such as in EmbedRank (Bennani-Smires et al., 2018) and Key2Vec (Mahata et al., 2018). In our work, we also rely on sentence encoders, but we use them to generate training data for sequence tagging. Therefore, we do not rely on sentence encoders at runtime to extract terminology from text, enabling the creation of lower latency systems.

To capture complex morphological structures we use word segmentation techniques. Word segmentation algorithms such as Byte-Pair Encoding (Sennrich et al., 2016), word-piece (Schuster and Nakajima, 2012), and unigram language modeling (Kudo, 2018) have been introduced to avoid the problem of out-of-vocabulary words and, more in general, to reduce the number of distinct symbols that sequence models for natural language processing have to process. To the best of our knowledge, we are the first to use the subword-unit tokenization as a signal to extract technical terms from text.

Our approach builds on the notion of specificity to find terminology. While there are multiple research works (Caraballo and Charniak, 1999; Ryu and Choi, 2006) highlighting the importance of specificity, to the best of our knowledge, this is the first work using the notion of specificity to extract terminology from text.

3 The approach

Figure 2 depicts our *weakly supervised* setup. Starting from a raw text corpus and no labels, our training workflow produces an efficient sequence tagging model, based on the transformer architecture, which effectively implements the term extraction. At the core of the *weak labels* there is a fully unsupervised component, called the *Unsupervised*

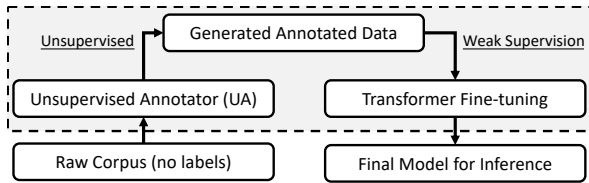


Figure 2: Our training workflow consists of 1) generating training data from raw unlabeled text using our Unsupervised Annotator, and 2) fine-tuning a transformer-based model or any sequence tagging model.

Annotator (UA), which, given the raw corpus, produces a training dataset for sequence labeling. The resulting dataset is used to train (or fine-tune) a sequence model that represents the final model for term annotation used at inference time. Pre-trained transformer-based models clearly represent a valid alternative to implement such sequence models. Moreover, we can avoid pre-training since the UA potentially generates a large amount of training data.

From the software engineering standpoint, this setup is extremely attractive as it makes the architecture of the term extraction subsystem modular and very flexible. The modularity comes from decoupling the inference component and the unsupervised annotator (UA). The unsupervised annotator can be enhanced with additional and more computationally demanding subcomponents (e.g., an entity linker to an external knowledge base), without increasing the final inference latency observed by the user. This modularity enables domain customization with proprietary data (and systems), which might be available for specific domains or customers. Since the integration between the Unsupervised Annotator and the inferencing component is achieved via data (i.e., the training samples for sequence tagging expressed in IOB format) the approach enables the smooth transition between a fully unsupervised setup and a setup where manual annotations augment the ones obtained via the UA. In practice, in realistic deployments, the unsupervised annotator is used to *bootstrap* the term extraction subsystem, while domain specific annotations are added over time by organizing annotation campaigns or by collecting labels through the interactions of the users with the knowledge discovery platform.

Having a dedicated component for inferencing, which is independent from the UA, gives the flexibility to select the right trade-off in terms of accuracy, inference latency, deployment costs, and

inferencing infrastructure. This choice is completely independent from the Unsupervised Annotator, which can be independently improved without taking care of inference latency. Since the inference component can be built around off-the-shelf transformer-based models, one can fully leverage the optimizations available in modern commercial offerings for inferencing services (e.g., Amazon Sagemaker, HuggingFace Infinity). As Transformer-based models are frequently used for multiple tasks (e.g., classification, NER, QA) within a knowledge discovery platform, this often corresponds to having a very homogeneous inferencing infrastructure in production. However, given that the UA can potentially generate a large amount of training samples, large pre-trained models are not a necessity, and even alternative architectures such as pQRNN (Kaliamoorthi et al., 2021) or pNLP-Mixer (Fusco et al., 2022) can be used.

3.1 Unsupervised annotator

Our unsupervised annotator is responsible for providing accuracy in potentially unseen domains *without* any training data, as depicted in Figure 1. It achieves this goal by using a greedy approach that processes each sentence of a raw corpus using the following steps:

- 1. Extract multiword expression candidates.** Using the part-of-speech tags we extract multiword expression candidates, consisting of sequences of zero or more adjectives (ADJ) followed by nouns (NOUN) or proper nouns (PROPNS) sequences. This chunking step allows us to identify term candidates expressed via multiword expressions.

- 2. Filter candidates by specificity or topic score.** Once the candidate terms, represented as multiword expression, are identified, a pruning step is responsible for filtering out multiword expressions using two semantic scores: the *topic score* and the *specificity score*. To compute those scores, we rely on pre-trained sentence encoders to extract embeddings from text.

- **Topic score.** The topic score captures the similarity, topic-wise, between a candidate and the sentence containing it. It is computed as the cosine similarity between the embedding vector of the multiword expression and the embedding vector of the sentence containing it.

- **Specificity score (SP).** This is the mean of the pairwise distance, in the embedding space, between the multiword expressions and all the other word or

multiword expression in the context. Specifically, given a multiword mw , and the word or multiword expression w_1, \dots, w_k in its context, we define the specificity score SP as:

$$SP(mw) = \frac{\sum_{i=1}^k dist(w_i, mw)}{k}, \quad (1)$$

where $dist(w_i, w_j)$ is the cosine-similarity between the embedding vectors of w_i and w_j . Multiword expressions with a higher score correspond to more specific terms.

Multiword expressions with a specificity or topic score below a certain threshold can be filtered out. Both scores rely on high-quality sentence encoders. In our implementation we use the pretrained sentence encoders described in Reimers and Gurevych (2019), but other sentence encoders can be used as a drop-in replacement.

3. Upgrade single nouns according to morphological features. At this stage, we could have nouns that are not part of any multiword expressions, but still relevant. We deal with those cases separately. For each of those nouns, we have to decide whether to extract them as terms or not. To do so, we use morphological features. First, we check if the lemma of the noun is the same as any of the heads of the multiword expressions. If that is the case, we upgrade the noun to term. Otherwise, we segment the word using a subword-unit segmentation algorithm and a vocabulary trained over a large general purpose corpus. Subword-unit tokenizers have been introduced to enable the representation of any text as a combination of subword units, with the idea that the most frequent words can be represented by a small number of subword units, eventually just one for very common words as in case for stopwords. For example, the word “*sun*”, will have its own entry in the dictionary of subword units, while the word “*paracetamol*” will be represented as the sequence of the following subword units: [“*para*”, “*##ce*”, “*##tam*”, “*##ol*”]. Not surprisingly, the number of subword units required to represent a word in a subword-unit tokenization regime is a very strong morphological signal, which we use as an *indirect measure* of the morphological “complexity”, and is extremely cheap to compute. In our implementation, we simply promote as terms all the nouns with a number of sub-tokens higher than a threshold (4 in our case). We use the vocabulary of the BERT-base model from HuggingFace (Wolf et al., 2020) and the corresponding tokenizer.

Corpus	Sentence			Terms		
	Train	Dev	Test	Train	Dev	Test
ACL	828	276	280	2,574	898	930
GENIA	11,127	3,709	3,710	48,928	16,217	16,404
ScienceIE	2,516	417	876	6,067	1,052	1,885

Table 1: Number of sentences and terms in the train, dev, and test set for the datasets used for evaluation.

4 Experiments

We now assess whether our approach can represent a *valid baseline* for term extraction in different technical domains when annotated data is *not available*. We aim to answer the following research questions:

- Does our Unsupervised Annotator generate a high-quality weakly-annotated dataset from a unlabeled general-domain corpus?
- Can we train models on the latter to lower the latency inference **and** increase the prediction performance at the same time?

4.1 Datasets

We use three common publicly available term extraction corpora: ACL RD-TEC 2.0 (QasemiZadeh and Schumann, 2016), GENIA (Kim et al., 2003), and ScienceIE (Augenstein et al., 2017). Each contains abstracts from scientific articles in different domains: natural language processing (ACL), medicine (GENIA), and computer science, material science, as well as physics (ScienceIE). All tokens are annotated using the IOB format (short for Inside, Out and Begin) (Ramshaw and Marcus, 1999). Since we are only interested in general term extraction, we did not use multiple class labels, even if provided in the respective dataset. We create random splits of train, dev, and test sets (60/20/20) for the ACL and GENIA datasets, and we use the pre-existing data splits for ScienceIE corpus.

In terms of preprocessing, we remove nested terms from the GENIA dataset, since the IOB tag set does not allow nested term extraction. For the ACL corpus, some samples have abstracts labeled by two annotators. In those cases, we selected the abstract from the first annotator. An overview of the datasets is given in Table 1.

Since our objective is to study the generalization of our approach, we need an *unlabeled* broad corpus from which our Unsupervised Annotator will annotate the text. Hence, we randomly sampled 500,000 sentences from abstracts from Semantic

Model (#Params)	ACL		Model (#Params)	GENIA		Model (#Params)	ScienceIE	
	exact F_1	partial F_1		exact F_1	partial F_1		exact F_1	partial F_1
BERT B (110M)	78.69	91.06	BERT B (110M)	70.13	88.19	BERT B (110M)	49.62	66.36
ELECTRA S (14M)	72.84	88.06	ELECTRA S (14M)	67.73	88.04	ELECTRA S (14M)	46.43	68.45
ELECTRA XS (7M)	50.40	71.61	ELECTRA XS (7M)	59.86	83.16	ELECTRA XS (7M)	27.17	51.10
UA (0)	49.95	74.56	UA (0)	45.65	77.16	UA (0)	39.75	64.29

Table 2: Results for the unsupervised annotator (UA) and transformer models fine-tuned on the **manually annotated** ACL, GENIA, and ScienceIE datasets, respectively. Without using any annotation, the UA performs similarly to ELECTRA XSmall and even better on the ScienceIE.

Scholar (SS).³ We call our weakly annotated training set UA-SS. The training sets of the ACL, GENIA, and ScienceIE datasets are not used (unless specified).

4.2 Models

We use transformer-models, fine-tuned with manual annotations, as baselines. We employ pre-trained transformer models of different sizes: BERT-base (110M parameters) (Devlin et al., 2019), ELECTRA Small (14M parameters) (Clark et al., 2020), and ELECTRA XSmall (7M parameters).

Since our main goal is to compare the models to each other and across multiple corpora, we prioritize comparability across corpora over comparability with approaches from other studies.

4.3 Experimental settings

We use the pre-trained checkpoints of BERT-base and ELECTRA Small from HuggingFace (Wolf et al., 2020). We pre-train ELECTRA XSmall⁴ from scratch using our Semantic Scholar dataset. During fine-tuning, we devoted a similar amount of GPU time to all the models. We pick the best-performing model in the dev set after 10 epochs

We implemented our Unsupervised Annotator using the POS tagger of SpaCy (Honnibal et al., 2020). To compute the specificity and similarity scores we use the sentence embedding model `distilbert-base-nli-mean-tokens` from the `sentence-transformers`⁵ library.

The specificity and similarity thresholds used to generate the training data over abstracts from Semantic Scholar have been set to conservative values. We set the threshold for the specificity $T_{SP} = 0.05$ and the threshold for the similarity

$T_{topic} = 0.1$. For the sub-word tokenization we rely on the tokenizer from BERT-base.

4.4 Results

In Table 2, we first compare the performance (expressed as exact and partial F_1 scores that count only exact or partial matches as true positives) of our fully *Unsupervised Annotator* to the performance obtained by fine-tuning transformer-based models with the manual annotations present in the original training sets. Without relying on any human annotation, our UA delivers comparable or even better results than the ELECTRA XSmall in ACL and ScienceIE, respectively. These results show that the UA represents a very *competitive baseline* for domains where annotations are not available.

Further, we are interested in understanding whether transformer-based models fine-tuned with human annotations can generalize across domains. We also evaluate if the availability of weakly supervised labels generated by our *Unsupervised Annotator* over a large and broad corpus (i.e., Semantic Scholar) could lead to models with higher generalization capabilities. In Table 3 we report the exact and partial F_1 scores for the ACL, GENIA, and ScienceIE datasets, and the transformer-based model fine-tuned with the output of our *Unsupervised Annotator* (UA-SS). This setup simulates the problem of bootstrapping an annotator for a specific domain for which in-domain human labels are not available.

On the ACL corpus, the UA-SS-based model clearly outperforms the GENIA-based and ScienceIE-based models. On the GENIA corpus, the UA-SS-based model and the ACL-based model perform equally well. On the ScienceIE corpus, all models perform equally with a slight tendency towards the GENIA-based model.

Overall, it can be said that the UA-SS-based approach is a valid starting point to bootstrap a

³www.semanticscholar.org/.

⁴We used 2 attention heads and 4 hidden layers, while using the same hidden dimension and similarly sized vocabulary.

⁵pypi.org/project/sentence-transformers/.

Model (#Params)	Fine-tuned on	ACL		GENIA		ScienceIE	
		exact F_1	partial F_1	exact F_1	partial F_1	exact F_1	partial F_1
BERT Base (110M)	UA-SS	58.22	77.36	53.18	79.38	46.79	66.59
	ACL	—	—	52.05	82.49	47.88	69.97
	GENIA	45.97	61.53	—	—	48.50	69.84
	ScienceIE	38.28	54.92	46.91	73.16	—	—
ELECTRA Small (14M)	UA-SS	58.00	77.41	53.44	80.01	44.68	65.58
	ACL	—	—	50.84	81.33	44.21	67.57
	GENIA	46.65	67.21	—	—	45.79	68.83
	ScienceIE	42.58	66.02	43.48	76.77	—	—
ELECTRA XSmall (7M)	UA-SS	49.83	72.78	45.35	74.83	40.32	62.39
	ACL	—	—	31.13	59.99	28.79	58.20
	GENIA	29.81	58.17	—	—	30.00	59.61
	ScienceIE	20.60	33.53	39.95	68.63	—	—

Table 3: Results for the generalization of multiple transformer models that are fine-tuned on the **weakly annotated** dataset based on the Semantic Scholar corpus (annotated with UA, denoted as UA-SS) and evaluated on the ACL, GENIA, and ScienceIE datasets, respectively. Transformer models fine-tuned using our automatically generated dataset perform better than their counterparts fine-tuned using the other datasets.

system in a no-resource scenario. Table 2 shows that the F1 score gap between models trained with in-domain manually annotated data and the UA-SS-based approach is lower for smaller models.

Now, we compare the *Unsupervised Annotator* with the models fine-tuned with its output to evaluate our two-step approach in terms of F1 score *and* inference latency. Figure 3 reports the average inference latency for models (fine-tuned with the UA-SS training data) over sentences from the ACL dataset with a batch of size 1 using a NVIDIA Tesla V100 and a single core of a Xeon E5-2690 v4 (similar trends on the other datasets). While the inference latency has similar orders of magnitude across models with GPU acceleration, the minimum inference time of 26.6 ms can be obtained on a *single CPU core* using the ELECTRA XSmall model. Therefore, our approach is particularly attractive in all cases where inference accelerators (e.g., GPUs) are not available. Additionally, the results highlight that by fine-tuning over the output of the UA, the latency can be *reduced by 4 to 10 times*, while providing comparable or even better F1 scores. Having the option to generate a large amount of training data for fine-tuning is an extremely useful property that enables the creation of very small models offering low inference times even without using GPU acceleration.

4.5 Lessons learned

In this work, we have demonstrated that, while the value of in-domain labels is without any doubt the best way to increase predictive quality, fully un-

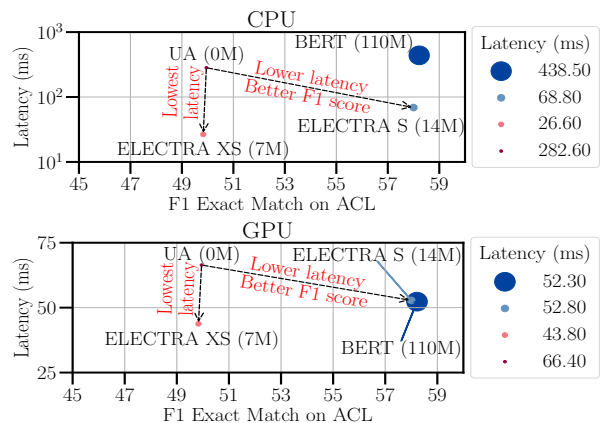


Figure 3: Average inference latency on CPU (top) and GPU (bottom) on the ACL dataset. We note in parenthesis the number of trainable parameters of the models. By fine-tuning over the output of the UA, we achieve lower latency and higher F1 scores. The lowest inference latency, 26.6 ms, is achieved on CPU.

supervised approaches are often the only viable option to bootstrap a term extractor that has to generalize across very diverse domains. Additionally, while the practicality of ML solutions is often underestimated, we have shown that having a modular system can not only provide greater flexibility in deployments, but can also allow to boost time predictive performance and inference latency at the same.

5 Conclusion

In this paper, we described an effective term extraction approach that uses a *fully unsupervised*

annotator to generate training data to fine-tune transformer models. This approach reduces the inference time of the unsupervised annotator, without decreasing its performance, and allows the flexibility to pick the right trade-off between latency and F1 score. The latency-optimized models are less than 30 Megabytes in size, provide inference latencies lower than 30 ms even *without* GPUs, while exhibiting a competitive F1 score compared to the models fine-tuned with manually annotated data.

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.
- Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. [Simple unsupervised keyphrase extraction using sentence embeddings](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 221–229, Brussels, Belgium. Association for Computational Linguistics.
- Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. 2005. *Ontology Learning from Text: Methods, Evaluation and Applications*.
- Sharon A. Caraballo and Eugene Charniak. 1999. [Determining the specificity of nouns from text](#). In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.
- Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. [Exclusive hierarchical decoding for deep keyphrase generation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.
- Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. [Pre-training transformers as energy-based cloze models](#). In *EMNLP*.
- 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 Dognin, Igor Melnyk, Inkit Padhi, Cicero Nogueira dos Santos, and Payel Das. 2020. [DualTKB: A Dual Learning Bridge between Text and Knowledge Base](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8605–8616, Online. Association for Computational Linguistics.
- Nazanin Firoozeh, Adeline Nazarenko, Fabrice Alizon, and Béatrice Daille. 2020. [Keyword extraction: Issues and methods](#). *Natural Language Engineering*, 26(3):259–291.
- Francesco Fusco, Damian Pascual, and Peter Staar. 2022. [pNLP-Mixer: an Efficient all-MLP Architecture for Language](#). *arXiv preprint arXiv:2202.04350*.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Ihab F. Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. [Saga: A platform for continuous construction and serving of knowledge at scale](#). In *Proceedings of the 2022 International Conference on Management of Data, SIGMOD/PODS '22*, page 2259–2272, New York, NY, USA. Association for Computing Machinery.
- Prabhu Kaliamoorthi, Aditya Siddhant, Edward Li, and Melvin Johnson. 2021. [Distilling large language models into tiny and effective students using pqrrn](#). *CoRR*, abs/2101.08890.
- J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. [Genia corpus—a semantically annotated corpus for bio-textmining](#). *Bioinformatics*, 19(suppl_1):i180–i182.
- Taku Kudo. 2018. [Subword regularization: Improving neural network translation models with multiple subword candidates](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.
- Juan Antonio Lossio-Ventura, Clément Jonquet, Mathieu Roche, and Maguelonne Teisseire. 2016. [Biomedical term extraction: overview and a new methodology](#). *Information Retrieval Journal*, 19(1-2):59–99.
- Dehong Ma, Sujian Li, Fangzhao Wu, Xing Xie, and Houfeng Wang. 2019. [Exploring sequence-to-sequence learning in aspect term extraction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3538–3547, Florence, Italy. Association for Computational Linguistics.
- Debanjan Mahata, John Kuriakose, Rajiv Ratn Shah, and Roger Zimmermann. 2018. [Key2Vec: Automatic ranked keyphrase extraction from scientific articles using phrase embeddings](#). 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 634–639, New Orleans, Louisiana. Association for Computational Linguistics.

- 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.
- 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.
- 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*. Association for Computational Linguistics.
- Pum-Mo Ryu and Key-Sun Choi. 2006. [Taxonomy learning using term specificity and similarity](#). In *Proceedings of the 2nd Workshop on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pages 41–48, Sydney, Australia. Association for Computational Linguistics.
- Antonio Šajatović, Maja Buljan, Jan Šnajder, and Bojana Dalbelo Bašić. 2019. [Evaluating automatic term extraction methods on individual documents](#). In *Proceedings of the Joint Workshop on Multiword Expressions and WordNet (MWE-WN 2019)*, pages 149–154, Florence, Italy. Association for Computational Linguistics.
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, 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.
- 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.
- Di Wu, Wasi Uddin Ahmad, Sunipa Dev, and Kai-Wei Chang. 2022. [Representation learning for resource-constrained keyphrase generation](#). *CoRR*, abs/2203.08118.
- Ziqi Zhang, Johann Petrak, and Diana Maynard. 2018. [Adapted textrank for term extraction: A generic method of improving automatic term extraction algorithms](#). *Procedia Computer Science*, 137:102 – 108. Proceedings of the 14th International Conference on Semantic Systems 10th – 13th of September 2018 Vienna, Austria.

DynaMaR: Dynamic Prompt with Mask Token Representation

Xiaodi Sun^{1,*}, Sunny Rajagopalan^{2,*}, Priyanka Nigam³, Weiyi Lu³, Yi Xu³
Iman Keivanloo³, Belinda Zeng³, Trishul Chilimbi³

¹Microsoft, ²Google, ³Amazon

¹xiaodisun315@gmail.com, ²sunny.rg@gmail.com

³{nigamp,weiyilu,yxaamzn,imanke,zengb,trishulc}@amazon.com

Abstract

Recent research has shown that large language models pretrained using unsupervised approaches can achieve significant performance improvement on many downstream tasks. Typically when adapting these language models to downstream tasks, like a classification or regression task, we employ a fine-tuning paradigm in which the sentence representation from the language model is input to a task-specific head; the model is then fine-tuned end-to-end. However, with the emergence of models like GPT-3, prompt-based fine-tuning has been proven to be a successful approach for few-shot tasks. Inspired by this work, we study discrete prompt technologies in practice. There are two issues that arise with the standard prompt approach. First, it can overfit on the prompt template. Second, it requires manual effort to formulate the downstream task as a language model problem. In this paper, we propose an improvement to prompt-based fine-tuning that addresses these two issues. We refer to our approach as DynaMaR – **D**ynamic **P**rompt with **M**ask **T**oken **R**epresentation. Results show that DynaMaR can achieve an average improvement of 10% in few-shot settings and improvement of 3.7% in data-rich settings over the standard fine-tuning approach on four e-commerce applications.

1 Introduction

Unsupervised pre-trained Language Models (LMs) such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) have achieved state-of-the-art performance on many natural language understanding tasks. In general, these models are fine-tuned for different tasks through the addition of a task-specific head on top of the [CLS] token representation (Scao and Rush, 2021).

An alternative method to applying LMs on downstream tasks is through discrete prompts. A discrete

prompt is an additional text phrase inserted along with the original input text that encapsulates the task of interest. By adding the prompt, we convert the downstream task into a masked language (MLM) problem. For example, to classify the sentiment of a movie review, “I hate this movie.”, we can append a prompt to the input to get “I hate this movie. It was [MASK]”. The pre-trained language model is thus prompted to identify the sentiment of the input statement and classify the [MASK] token as “terrible” instead of “great” (Liu et al., 2021). In this paper, we call a function that includes a prompt and its position information a prompt template.

Prompt-based approaches have shown success in low-data regimes (Petroni et al., 2019; Schick and Schütze, 2021; Jiang et al., 2020; Gao et al., 2021; Lester et al., 2021). Prompt-based fine-tuning is beneficial in few-shot learning, because it provides extra task information to the model through the prompt text (Schick and Schütze, 2021). However, when we explore this technique in practice, two issues have arisen. First, the trained model can overfit on words or phrases within the prompt and on the position of the [MASK] token in the prompt (Zhong et al., 2021). For example, in movie review sentiment analysis, when we append the prompt, “Does the user like the movie? [MASK]”, to a negative review, “This is a bad movie.”, the trained model is inclined to predict the positive class, because the word “like” frequently appears in positive reviews and the masked language model has greater attention on the words/phrases that are closer to the mask token as shown in Figure 1. We call this issue prompt-related overfitting in this work.

We tackle prompt-related overfitting by introducing a dynamic prompt approach. In this approach, we create a prompt pool consisting of multiple prompt templates. To construct this pool, we generate a set of prompt candidates and filter by a similarity score we propose, called the pairwise prompt dissimilarity score (detailed in Section 3). We then

*Work done while at Amazon.

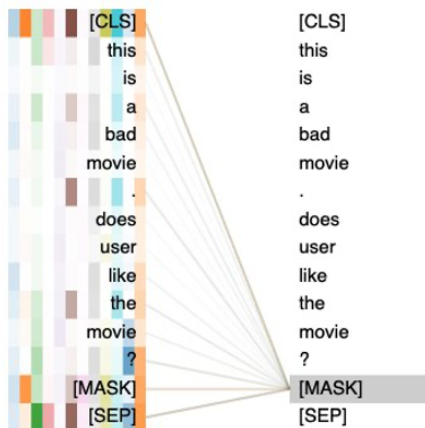


Figure 1: BERT Attention Distribution. The figure shows that the MLM model puts greater attention on the prompt than the original input.

introduce the dynamic component of the algorithm by randomly selecting a prompt template from the pool and applying to the input for each training step. For example, in the movie review sentiment analysis task, the trained model will randomly see either “does the user like the movie? [MASK]” or “does the user dislike the movie? [MASK]” appended to the original input. This prevents the model to overfit on spurious correlations between words in the prompt and the class label.

In addition, as previously mentioned, the standard prompt-based fine-tuning setup can be inefficient. It requires significant input and answer engineering to reformulate the downstream tasks as MLM problems (Liu et al., 2021). This process is time-consuming especially for tasks with large numbers of classes. Besides, another disadvantage of the standard setup is that it cannot be directly applied to regression problems, as they cannot be easily converted to MLM problems. To simplify this process, we fine-tune the model by feeding the mask token representation into a task-specific classifier/predictor head instead of the pre-trained MLM head to avoid the answer engineering process, as shown in Figure 2. We refer to our prompt-based approach with these two improvements as Dynamic Prompt with Mask Token Representation (DynaMaR). We apply DynaMaR to both few-shot and data-rich settings and, for the first time, show improvement gains across four tasks not only in few-shot settings but also in data-rich settings.

Our contributions include: (1) proposing DynaMaR, which can be applied without reformulating downstream tasks into language problems and is

robust to prompt-related overfitting, (2) showing DynaMaR can achieve improvements in both few-shot and data-rich settings, (3) proposing a prompt dissimilarity score to evaluate the degree of dissimilarity between two prompt templates and to help construct a diverse dynamic prompt pool, (4) demonstrating that a larger dynamic prompt pool achieves better performance on downstream tasks.

2 Related Work

Our work can be divided into three components: language model fine-tuning, prompt generation, and the design of the prompt template.

Language Model Fine-tuning is the main focus of our work. Recently, a large amount of research has focused on improved language model finetuning methods (Howard and Ruder, 2018; Dodge et al., 2020; Lee et al., 2020; Zhang et al., 2021). These works mainly focus on optimization and regularization techniques to stabilize fine-tuning. In contrast to these works, Gao et al. (2021) describe the concept of prompt-based fine-tuning for language models. We adapt and simplify the core ideas from this work to create a simple yet efficient prompt-based fine-tuning approach.

Prompt Generation is a key process in prompt-based fine-tuning. The choice of prompt significantly influences performance. The most natural way to generate prompts is through manual design. Petroni et al. (2019) employ manually generated prompts with ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) models. They evaluate on the LAMA (LAnguage Model Analysis) benchmark (Bordes et al., 2013; Nickel et al., 2016) without fine-tuning and conclude that the model is able to recall knowledge learned from the pre-training tasks. While manually crafting prompts is intuitive, creating effective prompts through manual effort requires time, experience, and expertise. To address this issue, a number of automatic prompt searching methods have been proposed. For example, Jiang et al. (2020) propose a data mining-based method that searches for a prompt based on the shortest path between the original inputs and answers. They also propose paraphrasing-based methods that take a seed prompt and paraphrase it into several semantically similar expressions. Gao et al. (2021) treat prompt generation as a text generation task and utilize T5, a sequence-to-sequence pretrained model, in the template search process. They generate templates by specifying the position to insert a prompt

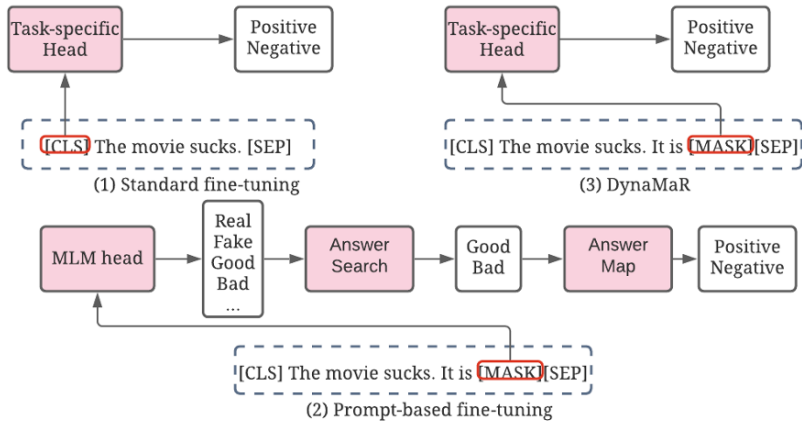


Figure 2: Fine-tuning approach demonstration.

template and then inputting samples into T5 to decode the templates. These automatic approaches achieve comparable performance to manually designed prompts. Besides, Logan IV et al. (2021) propose the null prompt method. Instead of generating prompts, they concatenate a [MASK] token with original inputs and it performs competitively to manually designed prompts. In our experiments, we utilize the prompt generation methods to create candidates for the dynamic prompt pool, while also including the null prompt approach as one of the baselines.

Prompt Template Design Factors are the factors that we take into consideration to create a metric that informs how prompts are selected for the dynamic prompt pool. Numerous previous works analyze prompt template design factors and the impact of prompt design on performance. Liu et al. (2021) summarize the factors that influence the application of prompt-related technologies in language models. Logan IV et al. (2021) note that the order in which the original input and the [MASK] token are concatenated is an important consideration. Zhong et al. (2021) propose to unify the prompts into a question-answering format. These previous works indicate that prompt construction impacts performance. To this end, we hypothesize that diversity in the set of prompt templates is an important factor in the performance of the model and propose a prompt dissimilarity score for measuring diversity.

3 Our Method: DynaMaR

In this section, we describe details of our approach, DynaMaR. Before explaining the training process, we define two concepts: the dynamic prompt pool and the inference prompt.

Dynamic Prompt Pool is a pool of prompt templates from which a prompt template will be randomly selected and applied to the input during training.

Inference Prompt is the prompt template used during inference. It is selected from the set of templates in the dynamic prompt pool. In general, it is the prompt template among those in the dynamic prompt pool that can achieve the highest performance in a fixed prompt setting.

We generate the candidates for the dynamic prompt pool and inference prompt through manual generation and paraphrasing-based methods proposed by Jiang et al. (2020). However, we do not include all candidates in the dynamic prompt pool. We want to ensure the prompts within a pool are sufficiently diverse so that the model will not overfit on any of them. Therefore, we introduce a prompt dissimilarity score to measure the level of dissimilarity between these candidates. We consider three factors in developing this metric: (1) prompt position, or whether to append or prepend the prompt to the input or even insert into the middle of pairwise inputs, (2) prompt wording or the prompt word selection, and (3) prompt format, or whether to create prompts in statement format or in the question-answering format proposed by Zhong et al. (2021). To define the prompt dissimilarity score, we first introduce the normalized Hamming

distance and the normalized Levenshtein distance.

Normalized Hamming Distance is equal to the number of different bits between two binary representations divided by the length of the binary representations (Norouzi et al., 2012). Let $HD(b_i, b_j)$ be the Hamming distance between binary representations b_i and b_j with equal length K . The equation of normalized Hamming distance $NHD(b_i, b_j)$ then follows:

$$HD(b_i, b_j) = \sum_{k=1}^K |b_{ik} - b_{jk}|, \quad (1)$$

$$NHD(b_i, b_j) = HD(b_i, b_j)/K. \quad (2)$$

Normalized Levenshtein Distance is equal to the minimum number of operations (substitution, insertion and deletion) required to transform a given string into another string divided by the length of the longer string and is calculated in a recursive fashion (Yujian and Bo, 2007). Let $LD(s_i, s_j)$ be the Levenshtein distance between string s_i and s_j . Let $|s_i|$ and $|s_j|$ be the length of prompt string s_i and s_j , respectively. Let $t(x)$ be a function that keeps a string of all but the first character of x . The equation of the normalized Levenshtein distance $NLD(s_i, s_j)$ follows:

$$LD(s_i, s_j) = \begin{cases} |s_i|, & \text{if } |s_i| = 0; \\ |s_j|, & \text{if } |s_j| = 0; \\ LD(t(s_i), t(s_j)), & \text{if } |s_i| = |s_j|; \\ 1 + \min \left(\begin{array}{l} LD(t(s_i), s_j), \\ LD(s_i, t(s_j)), \\ LD(t(s_i), t(s_j)) \end{array} \right), & \text{otherwise.} \end{cases} \quad (3)$$

$$NLD(s_i, s_j) = \begin{cases} \frac{LD(s_i, s_j)}{|s_i|}, & \text{if } |s_j| \leq |s_i|, \\ \frac{LD(s_i, s_j)}{|s_j|}, & \text{if } |s_i| < |s_j|. \end{cases} \quad (4)$$

Suppose we generate N prompt templates. Let p_i and p_j be two prompt templates with s_i, s_j as prompt strings, respectively, where $i \neq j$ and $i, j \in \{1, 2, \dots, N\}$. We treat the prompt position and format information as categorical variables and convert them into binary representations, b_i, b_j . Let $PDS(p_i, p_j)$ denote the prompt dissimilarity score between prompt templates p_i and p_j . The prompt dissimilarity score equation can be found below:

$$PDS(p_i, p_j) = NHD(b_i, b_j) + NLD(s_i, s_j). \quad (5)$$

In our experiment, we use 0.5 as the pairwise prompt dissimilarity score threshold. We add the prompt templates that have prompt dissimilarity

score larger than the threshold to others to a dynamic prompt pool. During the training process, we randomly pick one prompt template from the pool for each training step and apply it to the original input. We treat the mask token representation from the modified input as the sentence embedding and train the model by directly feeding it into a task-specific predictor head.

4 Experiment

4.1 Data

In this experiment, we use four e-commerce proprietary datasets: (1) Variation Elimination (VE), (2) Music Match (MM), (3) Music Genre (MG), and (4) Price Prediction (PP). VE is a binary classification problem with pairwise-document inputs where the label identifies whether two items are the variations of the same product or not. For example, similar shirts (from the same producer and brand) in different sizes or colors are considered to be variations. MM is a binary classification problem with pairwise-document inputs that identifies whether two music tracks from different sources are the same or not. MG is a 303-way classification problem with single-document inputs that classifies music tracks to genres. PP is a regression problem with single-document inputs that aims to estimate the sales price based on the product information. It should be noted that the percentage of inputs with number of tokens larger than 512 in VE, MM, MG, PP are 90%, 75%, 82%, 1%, respectively.

For each task, we split the dataset into three parts: (1) train, (2) validation, and (3) test. We use the full training dataset for the data-rich settings. We also sample multiple few-shot training datasets for few-shot learning settings. In few-shot learning, each classification dataset contains roughly 20 samples for each class. For the regression task (i.e., PP), we randomly sample 1% of the full training dataset as a few-shot training dataset.

4.2 Model and Tokenizer Setup

For training the tokenizer, we collect an English product catalog dataset with text features including title, description, and detail bullet points. We train a 32K BPE vocabulary on this dataset using the SentencePiece library (Kudo and Richardson, 2018).

We create a 500M parameter transformer encoder-only model, with 38 hidden layers, 1024 embedding size, 16 attention heads, and maximum

sequence length of 512. We train the model using the LANS optimizer (Zheng et al., 2020) with a batch size of 8192 and a learning rate of 10^{-4} on the product catalog dataset.

4.3 Prompt Generation and Selection

To create the dynamic prompt pool for our tasks, we first generate 20 prompt templates for each task and select 5 out of them using the prompt dissimilarity score. Specifically, for each task, we first manually design 10 prompt templates. By treating prompt template generation as paraphrase generation task (Jiang et al., 2020), we use these 10 prompt templates as seeds to generate another 10 templates per task by leveraging the public T5 paraphrase generation model from Hugging Face¹. Afterwards, we use the prompt dissimilarity score to select 5 prompt templates out of the 20 based on the method discussed at the end of Section 3. The selected prompt templates are used as each task’s dynamic prompt pool. For inference, we evaluate each template in the dynamic prompt pool through the evaluation process discussed in Section 4.5, and select the prompt template that produces the best performance on each task. Table 5 shows the dynamic prompt templates as well as the inference prompt selected for each task.

4.4 Fine-tuning (Ft) Methods

We compare DynaMaR with the following approaches:

- **Promptless Fine-tuning - CLS (PFt-CLS)** is our baseline approach where we fine-tune the model by feeding the [CLS] token representation into a predictor head.
- **Promptless Fine-tuning - Average Pooling (PFt-Avg)** fine-tunes the model by using the average of sequence output for prediction.
- **Null Prompt - Prefix (NP-Prefix)** prepends the [MASK] token to the original inputs and fine-tunes the model by feeding the [MASK] token representation into a predictor head. This approach avoids the issue where the model overfits the prompt template since it does not require a template.
- **Null Prompt - Suffix (NP-Suffix)** is the same as the above approach except that the [MASK]

¹https://huggingface.co/Vamsi/T5_Paraphrase_Paws

Ft Method	VE	MM	MG	PP	Avg
PFt-CLS	0	0	0	0	0
PFt-Avg	-1.5%	+7.2%	-3.7%	-8.8%	-1.7%
NP-Prefix	-1.0%	+4.1%	-2.0%	+2.6%	+0.9%
NP-Suffix	-2.6%	+0.2%	-1.6%	+6.7%	+0.7%
FiTeR	-0.7%	+13.9%	-1.1%	+7.3%	+4.9%
DPMR	+0.8%	+15.8%	-0.5%	+23.8%	+10.0%

Table 1: Few-shot Learning Performance Comparison.

token is appended to the inputs instead of being prepended.

- **Fixed Prompt with Mask Token Representation (FiTeR)** utilizes a static prompt template in both the training and inference stages and fine-tunes the model by feeding the [MASK] token representation into a predictor head.

Note that we use a task-specific predictor head in combination with all above approaches including the prompt-based fine-tuning methods, which typically use the pre-trained MLM head for prediction. The reason is that we have a regression task as one of our evaluation datasets, and as already discussed in Section 1, it is not straight forward to convert regression tasks into MLM tasks.

4.5 Model Training and Evaluation Setup

As mentioned in Section 1, we measure the performance in both few-shot and data-rich settings. For both VE and MM, we use Area Under the Precision-Recall Curve (PRAUC) as the evaluation metric. For MG, we use classification accuracy as the evaluation metric. For PP, we use Root Mean Square Error (RMSE) as the evaluation metric. We validate the performance every 2 training steps in the few-shot settings and every 100 steps in the data-rich settings. We use early stopping with a patience of 3 validation steps to select the best model for each task. We then evaluate the best models on the test datasets. For few-shot learning, we report the average performance across multiple few-shot datasets per task to reduce the variation in performance. In Table 1 and Table 2, we calculate and report the improvement percentage, which is the ratio of positive change as compared to PFt performance.

4.6 Results

Table 1 and 2 show the performance results for both few-shot and data-rich settings. In both settings, PFt-Avg shows degradation in average of

Ft Method	VE	MM	MG	PP	Avg
PfT-CLS	0	0	0	0	0
PfT-Avg	-0.1%	+1.2%	-1.0%	-11.0%	-2.7%
NP-Prefix	-0.1%	+1.0%	-0.4%	0	+0.1%
NP-Suffix	-0.3%	+1.7%	-0.7%	+2.2%	+0.7%
FiTeR	0	+1.5%	-0.2%	+3.3%	+1.2%
DPMR	0	+2.9%	-0.3%	+12.1%	+3.7%

Table 2: Data-rich Performance Comparison.

performance compared to PfT-CLS. This shows that average pooling generates worse sentence representations than does taking the [CLS] token representation.

In contrast, both null prompt approaches show improvement in average performance compared to PfT-CLS in both few-shot and data-rich settings. The improvement could be a result of aligning the format of the downstream tasks and that of the pre-training task. By changing the input format to be similar to that of the MLM task, we reduce the amount of data that are required to coach the model to learn the new task.

Also, there is a difference in the performance of NP-suffix and NP-prefix. This is likely due to the positional differences of the [MASK] token in the two methods. For example, suppose we want to perform sentiment analysis on a sentence like “I love the movie”. Prepending or appending the [MASK] token would result in different distances between [MASK] and the word “love”, which holds the key information for classification. Such positional differences could lead to different performance even though the two methods are very similar in spirit.

Another observation is that FiTeR shows higher improvement in average of performance compared to null prompt approaches. Recall that FiTeR introduces task information through the prompt templates, while the null prompt approaches do not, which supposedly addresses the issue where the model overfits the prompt templates. Hence, the results show that the benefits of adding the extra task information outweigh the possible performance loss caused by the prompt-related overfitting issue.

Finally, DynaMar outperforms FiTeR on all tasks in both setting, with the only exception being MG in the data-rich setting. This indicates that increasing the diversity of prompt templates used during training will improve model generalization. We also observe that DynaMar does not show significant improvement over PfT-CLS on both MG and VE. This is because both tasks contain a large num-

ber of documents with length longer than 512, as mentioned in Section 4.1. As a result of this, we need to truncate more of the original inputs for these tasks in order to insert prompts, which can lead to information loss. Thus, DynaMar is less efficient in problems with long documents.

4.7 Analysis

Larger dynamic prompt pool, better performance. The size of the dynamic prompt pool influences the performance. We compare the average improvement percentage across four tasks with the size of dynamic prompt pool = 1, 3, 5 (prompt information can be found in Appendix A). From Figure 3, we can see that performance improves as the dynamic prompt pool is made larger.

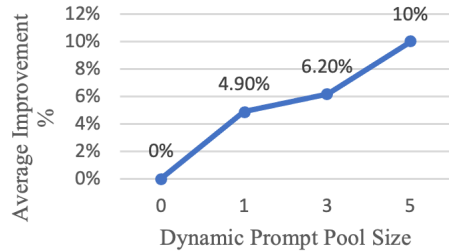


Figure 3: Pool Size vs Improvement Percentage.

4.8 Limitations and Future Directions

As mentioned in Section 4.6, our method does not show substantial improvement on tasks involving long documents. Besides, the threshold of prompt dissimilarity score can be treated as a parameter. This work lack of a study on the effect of this threshold. In addition, we focus on e-commerce related English classification/regression tasks in this work, the performance of our method in other nature language processing use cases remains unexplored. As a next step, we will conduct additional studies on these three topics.

5 Conclusion

In this work, we discuss methods for generating prompts and propose a way to select prompt templates to include in the dynamic prompt pool. Also, we show that using the mask representation of a prompt either equals or improves upon the performance of standard fine-tuning on four e-commerce applications in both few-shot and data-rich settings. In addition, we find DynaMaR outperforms the

fixed prompt approach in both settings. Furthermore, we show that a larger dynamic prompt pool leads to improved model performance when employing DynaMaR.

References

- Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- 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: Human Language Technologies (NAACL-HLT)*.
- Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. In *ArXiv*.
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.
- Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Association for Computational Linguistics (ACL)*.
- Zhengbao Jiang, Frank F. Xu, J. Araki, and Graham Neubig. 2020. How can we know what language models know? In *Association for Computational Linguistics (ACL)*.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *International Conference on Learning Representations (ICLR)*.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. In *ArXiv*.
- 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. In *ArXiv*.
- Robert L Logan IV, Ivana Balavzević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Maximilian Nickel, Kevin P. Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A review of relational machine learning for knowledge graphs. In *Proceedings of the IEEE*.
- Mohammad Norouzi, David J. Fleet, and Ruslan Salakhutdinov. 2012. Hamming distance metric learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.
- Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.
- Teven Le Scao and Alexander M. Rush. 2021. How many data points is a prompt worth? In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.
- Timo Schick and Hinrich Schütze. 2021. It’s not just size that matters: Small language models are also few-shot learners. In *Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT)*.
- Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*.
- Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample bert fine-tuning. In *International Conference on Learning Representations (ICLR)*.
- Shuai Zheng, Haibin Lin, Sheng Zha, and Mu Li. 2020. Accelerated large batch optimization of bert pretraining in 54 minutes. In *ArXiv*.

Ruiqi Zhong, Kristy Lee, Zheng Zhang, and Dan Klein. 2021. Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

A Dynamic Prompt Pool with Different Sizes

We need to define two prompt-related parameters while using DynaMaR: the dynamic prompt pool and the inference prompt. The list of prompts in the pool and the inference prompt selected for dynamic prompt pool sizes of 1, 3, and 5 can be found in Table 3, Table 4, and Table 5, respectively.

Task	Inference Prompt	Dynamic Prompt Pool
VE	$f(x_1, x_2) = x_1$ and x_2 are [MASK] product	$f(x_1, x_2) = x_1$ and x_2 are [MASK] product
MM	$f(x_1, x_2) = x_1$ and x_2 are [MASK] music	$f(x_1, x_2) = x_1$ and x_2 are [MASK] music
MG	$f(x) = \text{Genre: [MASK]} x$	$f(x) = \text{Genre: [MASK]} x$
PP	$f(x) = x$ The price is [MASK]	$f(x) = x$ The price is [MASK]

Table 3: Dynamic Prompt Pool Size = 1.

Task	Inference Prompt	Dynamic Prompt Pool
VE	$f(x_1, x_2) = x_1$ and x_2 are [MASK] product	(1) $f(x_1, x_2) = x_1 x_2$. Are they the same product? [MASK] (2) $f(x_1, x_2) = x_1$ and x_2 are [MASK] product (3) $f(x_1, x_2) = x_1 x_2$. They are [MASK]
MM	$f(x_1, x_2) = x_1$ and x_2 are [MASK] music	(1) $f(x_1, x_2) = x_1 x_2$. Are they the same song? [MASK] (2) $f(x_1, x_2) = x_1$ and x_2 are [MASK] music (3) $f(x_1, x_2) = x_1$ is as [MASK] as x_2
MG	$f(x) = \text{Genre: [MASK]} x$	(1) $f(x) = \text{Genre: [MASK]} x$ (2) $f(x) = \text{Music Genre: [MASK]} x$ (3) $f(x) = x$ what is genre of the music? [MASK]
PP	$f(x) = x$ The price is [MASK]	(1) $f(x) = \text{Price: [MASK]} x$ (2) $f(x) = x$ it cost [MASK] dollars (3) $f(x) = x$ what is price of the product? [MASK]

Table 4: Dynamic Prompt Pool Size = 3.

Task	Inference Prompt	Dynamic Prompt Pool
VE	$f(x_1, x_2) = x_1$ and x_2 are [MASK] product	(1) $f(x_1, x_2) = x_1 x_2$. Are they the same product? [MASK] (2) $f(x_1, x_2) = x_1$ and x_2 are [MASK] product (3) $f(x_1, x_2) = x_1 x_2$. They are [MASK] (4) $f(x_1, x_2) = \text{Are } x_1 \text{ and } x_2 \text{ the same product? [MASK]}$ (5) $f(x_1, x_2) = x_1$ is as [MASK] as x_2
MM	$f(x_1, x_2) = x_1$ and x_2 are [MASK] music	(1) $f(x_1, x_2) = x_1 x_2$. Are they the same song? [MASK] (2) $f(x_1, x_2) = x_1$ and x_2 are [MASK] music (3) $f(x_1, x_2) = x_1 x_2$. They are [MASK] music (4) $f(x_1, x_2) = \text{Are } x_1 \text{ and } x_2 \text{ the same music? [MASK]}$ (5) $f(x_1, x_2) = x_1$ is as [MASK] as x_2
MG	$f(x) = \text{Genre: [MASK]} x$	(1) $f(x) = \text{Genre: [MASK]} x$ (2) $f(x) = \text{Music Genre: [MASK]} x$ (3) $f(x) = x$ This is a [MASK] music (4) $f(x) = \text{Type: [MASK]} x$ (5) $f(x) = x$ what is genre of the music? [MASK]
PP	$f(x) = x$ The price is [MASK]	(1) $f(x) = \text{Price: [MASK]} x$ (2) $f(x) = x$ Price: [MASK] (3) $f(x) = x$ it cost [MASK] dollars (4) $f(x) = x$ The price is [MASK] (5) $f(x) = x$ what is price of the product? [MASK]

Table 5: Dynamic Prompt Pool Size = 5.

A Hybrid Approach to Cross-lingual Product Review Summarization

Saleh Soltan

Alexa AI, New York, USA
ssoltan@amazon.com

Victor Soto

Alexa AI, New York, USA
nvmartin@amazon.com

Ke Tran

Amazon AI Translate, Berlin, Germany
trnke@amazon.de

Wael Hamza

Alexa AI, Dallas, USA
waelhamz@amazon.com

Abstract

We present a hybrid approach for product review summarization which consists of: (i) an unsupervised extractive step to extract the most important sentences out of all the reviews, and (ii) a supervised abstractive step to summarize the extracted sentences into a coherent short summary. This approach allows us to develop an efficient cross-lingual abstractive summarizer that can generate summaries in any language, given the extracted sentences out of thousands of reviews in a source language. In order to train and test the abstractive model, we create the Cross-lingual Amazon Reviews Summarization (CARS) dataset which provides English summaries for training, and English, French, Italian, Arabic, and Hindi summaries for testing based on selected English reviews. We show that the summaries generated by our model are as good as human written summaries in coherence, informativeness, non-redundancy, and fluency.

1 Introduction

Summarizing product reviews with thousands of reviews is a daunting task. At the same time since this task is extremely time consuming to be done by humans, there are no annotated training datasets available for it. Hence, almost all existing approaches rely on unsupervised methods such as Latent Semantic Analysis (LSA) (Steinberger and Jezek, 2004), LexRank (Erkan and Radev, 2004a), MeanSum (Chu and Liu, 2019), and CopyCat (Bražinskis et al., 2020b) to name a few. However, the main two shortcomings of these methods are: (i) they do not provide comparable summaries in terms of coherency and fluency to human written summaries (Bražinskis et al., 2020a), and (ii) they cannot be used for cross-lingual summarization (i.e., provide summaries in a target language given the summaries in the source language).¹

¹Although it is possible to machine translate the summaries, it will add an extra inference time and may reduce the

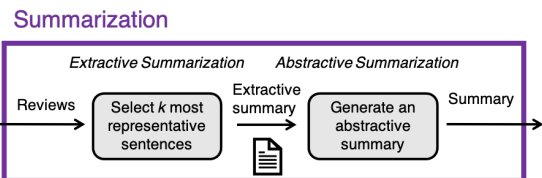


Figure 1: Our proposed summarization pipeline for product review summarization.

In order to address both of these shortcomings, we propose a hybrid two-step summarization approach: (i) an unsupervised extractive summarization step to extract the most representative sentences out of all the reviews, and (ii) a supervised abstractive summarization step to summarize the extracted sentences into a coherent short summary.

The main advantage of this approach is that we can rely on a light-weight unsupervised method for the extractive step to reduce the number of reviews and focus on a more expensive supervised model for the abstractive part (and actually collect human summaries for training it). Moreover, we can use state-of-the-art multilingual transformer models to develop a cross-lingual abstractive summarizer relying only on a monolingual extractive step.

For extractive summarization, we use LSA (Steinberger and Jezek, 2004) which is computationally efficient and provides a good performance in our application. To create training data for the abstractive step, we select 1000 products, for each product extract 10 most informative sentences out of all the English reviews using LSA, and obtain 3 summaries per product by asking Amazon Mechanical Turkers to write a short summary of the provided 10 sentences in English (further details are provided in Section 3). An example datapoint is provided in Table A1 in the Appendix.

For testing, we repeat the same process for another 280 products, but this time we ask Turkers to quality by translating name of the products etc.

write summaries in French, Spanish, Italian, Arabic, and Hindi (in addition to English) based on the selected English sentences (3 summaries per language per product). We name this dataset Cross-lingual Amazon Reviews Summarization (CARS) dataset.

Finally, we train multiple transformer based models on CARS training data and evaluate their performance on the test data through both automatic and human evaluations. We show that our approach provides informative summaries that are as good as human written summaries and can generalize well to the categories not in the training data (an example of a summary generated by our model is shown in Table A1 in the Appendix).

The main contributions of our work are two fold: 1) introducing a scalable and productionalizable approach for cross-lingual product review summarization capable of producing summaries in English, French, Spanish, Italian, Arabic, and Hindi from English reviews, and 2) demonstrating that our abstractive summarization model outperforms state-of-the-art opinion summarization method (Bražinskas et al., 2020a).

2 Related Work

Some of the most widely used extractive summarization techniques are based on Latent Semantic Analysis (LSA) (Dumais, 2004; Gong and Liu, 2001), in which a matrix that represents the importance of words in sentences is created, and singular value decomposition is used to select the sentences with the highest relevancy; or Bayesian Topic Modeling (Daumé III and Marcu, 2006; Haghighi and Vanderwende, 2009), in which a generative model is used to represent documents as mixtures of latent topics, where a topic is a probability distribution over words. Other extractive methods make use of graph methods (Erkan and Radev, 2004b) and machine learning methods (Wong et al., 2008; Narayan et al., 2018).

Abstractive summarization methods can be structure-based, in which a data structure is used to generate the new summary (examples include tree-based methods (Knight and Marcu, 2000; Kikuchi et al., 2014), template-based methods (Cao et al., 2018) and rule-based methods). More recently, the use of encoder-decoder architectures in transfer learning frameworks have facilitated the use of pre-trained encoders to generate document representations which are then used to generate a new

summary (Rush et al., 2015; Chopra et al., 2016; Liu and Lapata, 2019).

Since then there has been increasing efforts to tackle many of the challenges posed by this task: Nallapati et al. (2016) adds linguistically motivated embeddings and pointer networks to deal with out-of-vocabulary words; Paulus et al. (2018) adds reinforcement learning to the training objective to lessen exposure bias; Cohan et al. (2018) proposes a hierarchical model to encode the discourse structure research papers and an attention-based decoder to generate the summaries; Gehrmann et al. (2018) improves content selection performance on neural summarizers by incorporating an extra attention step to constrain on more likely phrases; Desai et al. (2020) incorporates two transformer models to predict the saliency and plausibility of sentence deletion in compressive summarization; Mao et al. (2020) introduces token-level constraints to improve factual consistency. To improve faithfulness Dou et al. (2021) proposes GSUM, a general guided summarization framework that can make use of external guiding policies to ensure faithfulness to the source documents.

On the topic of cross-lingual summarization, Chi et al. (2020) proposes a pre-training strategy for natural language generation tasks on both monolingual and multi-lingual settings followed by monolingual fine-tuning on the downstream task, and show that the resulting model can generalize to new languages. Ouyang et al. (2019) trains low resource cross-lingual summarization systems on automatically translated input and clean references, improving on a standard copy-attention summarizer on low resource languages and also evaluates on an unseen language.

The task of opinion or review summarization, which this paper focuses on, is receiving increased attention due to real-world practical usage. Bražinskas et al. (2020a) proposes FewSUM, a hierarchical framework for few-shot multi-document review summarization that consists on a transformer-based generator followed by plug-in network that switches the generator into a summarizer. FewSUM is the main system we will use for comparison throughout this paper. In a recent concurrent work (Bražinskas et al., 2021), the authors create a dataset of product summaries from a set of product reviews and propose to use joint learning to select a subset of reviews and then summarize from them. However, they use professional written summaries

from various websites as gold summaries (which may not be based on customer reviews at all, leading to model hallucination). The main advantage of our approach in production is its decoupled design. Namely, the extractive part of our system can always be improved to extract more informative sentences without a need to retrain the abstractive summarizer.

Finally, [Gamzu et al. \(2021\)](#) proposes the task of extreme summarization from multiple product reviews by extracting a single sentence that is concise, relevant and supported by multiple reviews. In our work, we propose to extract the most relevant sentences from each set of reviews, rank them, and use the top 10 reviews to create product summaries. Nevertheless, our decoupled design allows us to use [Gamzu et al. \(2021\)](#)’s method as our extractive summarizer and improve our end to end system in the future.

3 CARS Dataset

In this section, we describe the steps in creating the Cross-lingual Amazon Reviews Summarization (CARS) dataset.

3.1 Products Selection

3.1.1 Train

In order to have a diverse set of products, we selected 1000 products from Electronics, Beauty and Personal Care, Sports, Office Products, and Kitchen categories (200 each) from all of the products with more than 1000 English reviews in the Amazon US marketplace. In each category, we selected 100 products with the average score greater than or equal to 4 out of 5, and 100 products with the average score less than 4 out of 5 (since low-rated products do not have many reviews, we had to use 4 out of 5 threshold to separate well-reviewed products from not so well reviewed products).

3.1.2 Test

For test, we selected 280 products from Electronics, Beauty and Personal Care, Sports, Office Products, Kitchen, Apparel, Furniture, Lawn & Garden categories (40 each) with more than 1000 English reviews in the US marketplace (we added 3 extra categories compared to the training set to evaluate generalization ability of models). In each category, we selected 20 products with average score greater than or equal to 4 out of 5, and 20 products with average score less than 4 out of 5.

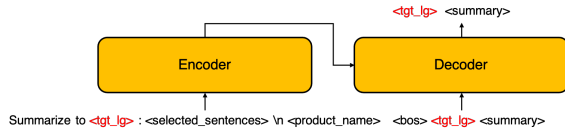


Figure 2: Training MBART50 for abstractive summarization. `<tgt_lg>` is the target language specific token.

3.2 Extract Sentences

Since products on Amazon have thousands of reviews, it is practically impossible to obtain a gold summary of all reviews for a product. Hence, we extracted a few sentences (out of all reviews) that best describe the important features of a product using an unsupervised extractive summarization method. In particular, we used Latent Semantic Analysis (LSA) ([Steinberger and Jezek, 2004](#)) which can run efficiently on reviews of products using a randomized version of the Singular Value Decomposition (SVD) to extract the top 10 sentences representing the reviews for each product (hyperparameter settings are provided in Appendix B). An example of the sentences extracted by LSA is provided in Appendix D.

3.3 Collect Human Summaries

We collected 3 human written summaries for each product in the training data in English and each product in the test data in English, French, Spanish, Italian, Arabic, and Hindi using Amazon Mechanical Turk (AMT). For each product, we provided the 10 selected sentences from all reviews along with the name of the product and asked Turkers to write a short summary of the selected sentences in the target language not exceeding 500 characters. More details on instructions and quality control of the collected summaries is provided in Appendix A.

4 Abstractive Summarizer

4.1 2-Step Training

To train an abstractive summarizer that can generate summaries in any of the target languages given a selected set of sentences in English, we used MBART50 ([Liu et al., 2020](#); [Tang et al., 2020](#)) model (which has already been fine-tuned on any-to-any translation task covering 50 languages including the ones presented in the CARS test set) and fine-tuned it for summarization task in 2 steps:

(a) Pre-fine-tuning: We used CNN-DailyMail dataset ([Hermann et al., 2015](#)) which includes 290K articles and the corresponding highlights in

English and machine translated all the highlights to Arabic, Hindi, and Italian. Moreover, we used ML-SUM dataset (Scialom et al., 2020) which provides 260K article-highlights pair in Spanish and 390K article-highlights pair in French, and machine translated the articles of these datasets from Spanish and French to English. That resulted in a multilingual dataset of English articles to all target languages highlights. We then extracted 10 sentences from each article using LSA to form a dataset of extracted sentences-highlight pairs that mimics the final task. Finally, we fine-tuned the MBART50 on this dataset as shown in Fig. 2.

(b) Fine-tuning: In the second stage of fine-tuning, we fine-tuned the model on CARS training data. Since the training data is in English only, we Machine Translated (MT) the summaries into Spanish, French, Italian, Hindi, and Arabic using Amazon Translate² to obtain a cross-lingual dataset from English sentences to summaries in the target languages. We then shuffled the input sentences to obtain two extra versions of the training data (total of 9000 sentences-summary pair for each language). We then fine-tuned the model on the combined dataset of all languages as shown in Fig. 2 with a small addition that we also add the name of the product after the selected sentences as the input to the encoder during fine-tuning. Appendix C provides fine-tuning hyper-parameter details.

4.2 Importance of Including Target Language Token on the Encoder Side

We observed an important property when fine-tuning MBART50. Despite the way MBART50 is fine-tuned for translation by only including the target language token on the decoder side, in our use case, we observed that relying only on the decoder to generate a summary in the target language from a unified representation of the selected sentences (provided by the encoder) significantly degrades the model performance (see Table 1). Hence, we included the target language token on the encoder side as well when we fine-tuned MBART50 for cross-lingual summarization task (as shown in Fig. 2).

5 Evaluation

All the models are trained and evaluated using HuggingFace Transformers Toolkit (Wolf et al., 2020).

²<https://aws.amazon.com/translate>

Model	EN			
	R1	R2	RL	BS
with <tgt_lg> in input	37.95	14.56	26.41	0.8832
w/o <tgt_lg> in input	16.34	6.07	11.91	0.8111

Table 1: The importance of including the target language token (<tgt_lg>) in the input (as shown in Fig. 2) when fine-tuning the model on cross-lingual summarization data. **R1** denotes Rouge-1, **R2** denotes Rouge-2, **RL** denotes Rouge-L, and **BS** denotes BERTScore.

5.1 Automatic Evaluation

For evaluation, we included two extra shuffled input sentence orders per gold summary in the test data per language to have a better estimate of the models’ performance (9 sentences-summary pairs per product). As in training, we added the name of the product to the end of the selected sentences as the input to the encoder. For evaluation metrics, we use Rouge (Lin, 2004) which is the most common metric for summary evaluation, and BERTScore (Zhang et al., 2020) which has been recently shown to correlates with human judgments the most. For BERTScore, we utilize RoBERTa (Liu et al., 2019) for English and mBERT (Devlin et al., 2019) for non-English summaries (which are the default choices).

Table 2 provides the cross-lingual summarization results on CARS dataset. As can be seen, adding machine translation of English summaries (notice that extracted sentences remain in English) to the training data gives a significant boost to models’ performance especially for non-English languages. Moreover, *pre-fine-tuning* the model on a similar task using public data can improve model performance especially in zero-shot case (i.e., training the model only on English summaries). A sample generated summary using the best model (last row in Table 2) is provided in Appendix D.

We also observe that adding machine translated Arabic and Hindi summaries to the training data decreases model performance in Rouge score on these languages (compared to zero-shot with pre-fine-tuning). The main reason for this degradation is that the MT translates the names of the products into Arabic and Hindi (and therefore model learns to translate them as well), whereas gold summaries have the names in English. However, this degradation is not present in BERTScore which relies on token representations instead of their face value.

To see how well the best model (last row in Table 2) generalizes to new categories (ones not in the

Model	EN				FR				ES			
	R1	R2	RL	BS	R1	R2	RL	BS	R1	R2	RL	BS
<i>Fine-tune only on public data</i>												
MBART50	24.28	7.36	18.46	0.8557	17.13	3.11	11.6	0.6506	15.88	2.63	11.76	0.6534
<i>Fine-tune on all English training data</i>												
MBART50	35.97	13.49	25.5	0.8779	8.04	3.84	7.21	0.6731	7.08	2.97	6.61	0.6778
+ pre-fine-tuning	35.92	13.49	25.52	0.876	12.64	5.15	10.35	0.6756	6.81	2.72	6.36	0.6758
<i>Fine-Tune on all English training data plus translation of summaries in all other languages</i>												
MBART50	37.51	14.17	26.21	0.8824	33.1	9.18	20.72	0.7134	34.19	8.64	21.6	0.7191
+ pre-fine-tuning	37.95	14.56	26.41	0.8832	33.4	9.21	20.73	0.7162	34.42	8.71	21.83	0.722
<i>Fine-tune only on public data</i>												
<i>Fine-tune on all English training data</i>												
MBART50	17.52	3.22	13.2	0.6433	11.4	4.87	11.27	0.6317	4.81	1.41	4.73	0.609
MBART50	6.94	2.93	6.26	0.6692	10.23	6.66	9.89	0.6332	6.01	3.37	5.79	0.6086
+ pre-fine-tuning	18.83	5.24	13.93	0.6805	24.65	14.95	24.18	0.6603	12.12	5.4	11.89	0.627
<i>Fine-Tune on all English training data plus translation of summaries in all other languages</i>												
MBART50	29.26	7.43	19.82	0.7119	20.11	9.07	19.8	0.7019	8.8	2.4	8.72	0.6655
+ pre-fine-tuning	29.31	7.42	19.7	0.7137	18.38	8.45	18.0	0.6995	8.87	2.62	8.83	0.6682

Table 2: Cross-lingual summarization results on the CARS set data averaged over 3 gold summaries and 3 different input sentence orders per product (9 samples per product). **R1** denotes Rouge-1, **R2** denotes Rouge-2, **RL** denotes Rouge-L, and **BS** denotes BERTScore.

training data), we computed per category performance of the model as well (see Table 3). Overall, we did not observe any particular degradation in our model’s performance on the categories that did not appear in the training data. This indicates that our model generalizes well across domains.

Category	EN			
	R1	R2	RL	BS
<i>Categories that appeared in the training data</i>				
Electronics	33.52	11.54	22.77	0.8757
Beauty	33.06	11.37	23.72	0.8778
Office Product	34.73	12.95	24.13	0.8764
Sports	40.79	17.55	28.69	0.8889
kitchen	41.23	16.92	28.05	0.8848
<i>Categories that did not appear in the training data</i>				
Lawn and Garden	38.97	15.01	27.62	0.885
Furniture	41.55	16.06	28.57	0.8895
Apparel	38.88	14	26.39	0.8849

Table 3: Per category performance of our best model.

5.2 Human Evaluation

To evaluate the summaries generated by our best model (last row in Table 2) by humans, we asked Turkers to compare two summaries (the gold summary from test set and the automatic summary by our model) in four different aspects: coherence, informativeness, non-redundancy, and fluency.

Since a high quality evaluation requires constant monitoring of Turkers, we managed to obtain evaluation for only the full English test set and the 2/3rds of the Spanish test set. We evaluated the results using the Best Worst Scaling (BWS) (Kiritchenko and

Feature	EN		ES	
	Binary	Multi	Binary	Multi
Coherence	-0.0108	0.0487	-0.0446	-0.0694
Informativeness	0.0195	0.1234	-0.038	-0.0992
non-redundancy	0.1061	0.1266	-0.0777	-0.1636
Fluency	0.1450	0.2792	-0.0645	-0.1537

Table 4: BWS scores based on human evaluation of the summaries. Binary BWS denotes the scores aggregated in the standard BWS way (-1 if the human summary is better, and +1 if the automatic summary is better) and Multi BWS denotes scores ranging between -3 to +3.

Mohammad, 2016) scores as presented in Table 4. As can be seen, for both English and Spanish the generated summaries are as good as human written summaries. Surprisingly, Turkers found generated summaries to be much better in Fluency in English compared to human written ones. Although the trend reverses in the Spanish summaries which is expected since our model relies only on machine translated training data for Spanish.

We also looked at the distribution of the scores on both languages (as presented in Figs. 3 and 4). For English, the automatic summaries are always rated to be “much better” more often than the human summaries, whereas for Spanish the opposite is true. In general Spanish summaries, whether automatic or crowdsourced, tend to rate their scores in the “a bit better” area, especially for informativeness, non-redundancy and fluency.

To see how good the generated summaries reflect the overall consensus over a product, we also asked Turkers to give the products a score from 1

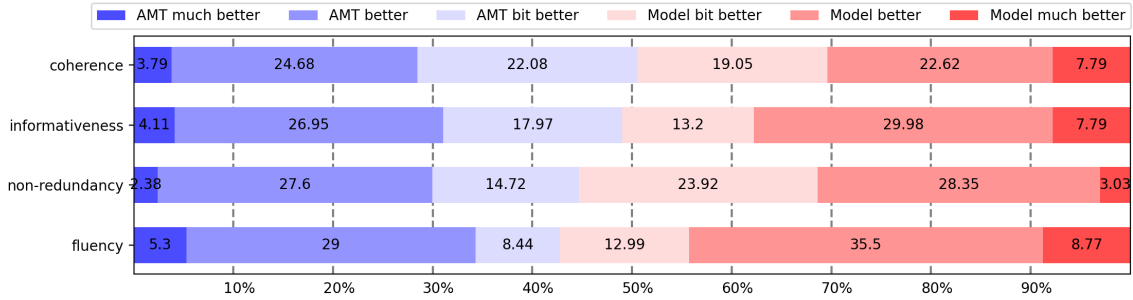


Figure 3: Human comparison between English gold summaries (AMT) and the ones generated by our best model.

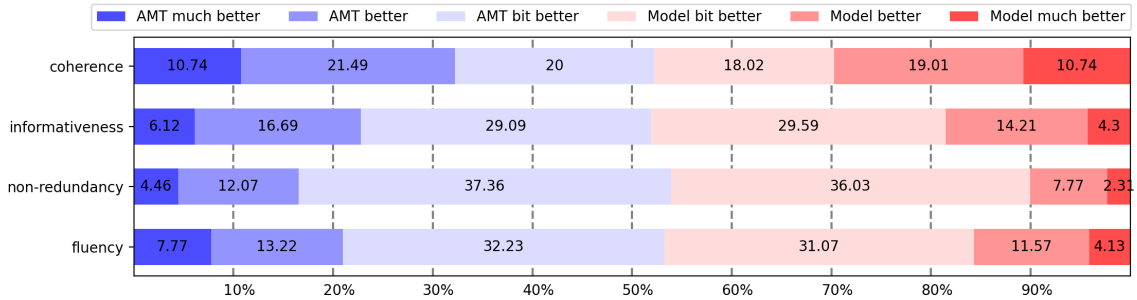


Figure 4: Human comparison between Spanish gold summaries (AMT) and the ones generated by our best model.

Model	R1	R2	RL
FewSum (Bražinskas et al., 2020a)	33.56	7.16	21.49
MBART50	30.12	8.02	19.32
+ pre-fine-tuning	36.01	8.71	23.10

Table 5: Rouge scores on the Amazon reviews test dataset provided in FewSum work. All the models are only fine-tuned on the training data of the same dataset.

to 5 based on the given reviews. We then computed the Mean Absolute Error (MAE) between the real amazon product scores and the guessed scores by the workers from the review summaries. For English, the MAE of both the human and generated summaries are very similar: 0.94 and 0.93, respectively. For Spanish the MAE of the crowdsourced summaries is slightly smaller than the MAE of the generated ones (0.62 compared to 0.75).

5.3 Comparison to FewSum

In the FewSum work (Bražinskas et al., 2020a), authors introduced an unsupervised pretraining strategy specifically for opinion review summarization using hundreds of thousands reviews and demonstrated that their model can provide state-of-the-art summaries only using few supervised review summaries (48 products each having 3 summaries based on 8 randomly selected reviews). However, as can be seen in Table 5, using MBART50 along with our "pre-fine-tuning" strategy (as described in Section 4), our model outperforms FewSum using

the exact same training data. Moreover, our model can generate summaries in other languages as well, using only English reviews.

6 Conclusion

We introduced a hybrid approach to cross-lingual product review summarization which provides summaries on different target languages by only relying on English reviews. We demonstrated that our approach results in review summaries that are as good as human written ones in English and Spanish (and comparable to gold summaries in other languages based on automatic evaluation metrics).

We also showed that our pre-fine-tuning plus fine-tuning approach can outperform state-of-the-art in few-shot abstractive review summarization. Moreover, since our abstractive summarizer is trained on summarizing a few selected (maybe unrelated) sentences, our end to end system can be improved by improving the extractive summarization component only without retraining the more expensive multilingual encoder-decoder architecture that we used for abstractive summarization (which is very desirable and cost saving feature in production systems).

The main shortcoming of our work is that it does not provide a mechanism to evaluate the correctness of the generated summaries which is part of our future work.

Acknowledgements

We thank Amjad Jbara and Tobias Falke for their helpful comments and suggestions. We also thank Amjad Jbara, Enrico Piovano, Nicolas Guenon Des Mesnards, and Varun Kumar for their help in data collection and reviewing the summaries written by Turkers in Arabic, Italian, French, and Hindi.

References

- Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020a. [Few-shot learning for opinion summarization](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4119–4135, Online. Association for Computational Linguistics.
- Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2020b. [Unsupervised opinion summarization as copycat-review generation](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5151–5169, Online. Association for Computational Linguistics.
- Arthur Bražinskas, Mirella Lapata, and Ivan Titov. 2021. [Learning opinion summarizers by selecting informative reviews](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9424–9442, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. [Retrieve, rerank and rewrite: Soft template based neural summarization](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, Melbourne, Australia. Association for Computational Linguistics.
- Zewen Chi, Li Dong, Furu Wei, Wenhui Wang, Xian-Ling Mao, and Heyan Huang. 2020. [Cross-lingual natural language generation via pre-training](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7570–7577.
- Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. [Abstractive sentence summarization with attentive recurrent neural networks](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California. Association for Computational Linguistics.
- Eric Chu and Peter J. Liu. 2019. [Meansum: A neural model for unsupervised multi-document abstractive summarization](#). In *Proc. ICML'19*.
- Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. [A discourse-aware attention model for abstractive summarization of long documents](#). 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 615–621, New Orleans, Louisiana. Association for Computational Linguistics.
- Hal Daumé III and Daniel Marcu. 2006. [Bayesian query-focused summarization](#). In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 305–312, Sydney, Australia. Association for Computational Linguistics.
- Shrey Desai, Jiacheng Xu, and Greg Durrett. 2020. [Compressive summarization with plausibility and salience modeling](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6259–6274, 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.
- Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. [GSum: A general framework for guided neural abstractive summarization](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4830–4842, Online. Association for Computational Linguistics.
- Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230.
- Günes Erkan and Dragomir R. Radev. 2004a. [Lexrank: Graph-based lexical centrality as salience in text summarization](#). *J. Artif. Int. Res.*, 22(1):457–479.
- Günes Erkan and Dragomir R. Radev. 2004b. [Lexrank: Graph-based lexical centrality as salience in text summarization](#). 22(1):457–479.
- Iftah Gamzu, Hila Gonen, Gilad Kutiel, Ran Levy, and Eugene Agichtein. 2021. [Identifying helpful sentences in product reviews](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 678–691, Online. Association for Computational Linguistics.
- Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. [Bottom-up abstractive summarization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*,

- pages 4098–4109, Brussels, Belgium. Association for Computational Linguistics.
- Yihong Gong and Xin Liu. 2001. [Generic text summarization using relevance measure and latent semantic analysis](#). In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*, page 19–25, New York, NY, USA. Association for Computing Machinery.
- Aria Haghighi and Lucy Vanderwende. 2009. [Exploring content models for multi-document summarization](#). In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370, Boulder, Colorado. Association for Computational Linguistics.
- K. Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and P. Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. NIPS'15*.
- Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. [Single document summarization based on nested tree structure](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 315–320, Baltimore, Maryland. Association for Computational Linguistics.
- Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *arXiv:1412.6980*.
- Svetlana Kiritchenko and Saif M. Mohammad. 2016. Capturing reliable fine-grained sentiment associations by crowdsourcing and best-worst scaling. In *Proc. NAACL-HLT'16*.
- Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710.
- 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.
- 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:1907.11692*.
- 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.
- Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. [Multilingual denoising pre-training for neural machine translation](#). *Transactions of the Association for Computational Linguistics*, 8:726–742.
- Yuning Mao, Xiang Ren, Heng Ji, and Jiawei Han. 2020. Constrained abstractive summarization: Preserving factual consistency with constrained generation. *arXiv:2010.12723*.
- Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016. [Abstractive text summarization using sequence-to-sequence RNNs and beyond](#). In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.
- 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.
- Jessica Ouyang, Boya Song, and Kathy McKeown. 2019. [A robust abstractive system for cross-lingual summarization](#). 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 2025–2031, Minneapolis, Minnesota. Association for Computational Linguistics.
- Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. *ICLR*.
- Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. [A neural attention model for abstractive sentence summarization](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.
- Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. [MLSUM: The multilingual summarization corpus](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8051–8067, Online. Association for Computational Linguistics.
- Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM'04*.
- Y. Tang, C. Tran, X. Li, P. Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pre-training and finetuning. *arXiv:2008.00401*.

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.

Kam-Fai Wong, Mingli Wu, and Wenjie Li. 2008. [Extractive summarization using supervised and semi-supervised learning](#). In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 985–992, Manchester, UK. Coling 2008 Organizing Committee.

Tianyi Zhang, V. Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proc. ICLR'20*.

Appendix

A Data Collection Details

A.1 Instructions

We asked Turkers to follow the following instructions when writing a summary for each product:

1. Summaries written in any language other than the asked language will be rejected.
2. Summaries that do not simplify very long product names will be REJECTED. For example, instead of referring to a hypothetical product as “FakeBrand FakeCode KN95 Protective Mask” you could use “FakeBrand KN95 Mask.”
3. Summaries written in first person will be REJECTED. Please write the summary on third person, and never in the first person.
4. Summaries that include information related to shipping or delivery will be REJECTED. Please do not include information related to the shipping or delivery of the product.
5. Summaries that include information related to other products will be REJECTED. Sometimes reviewers might compare the product of interest to other options in the market. It is ok to include how the product of interest fares compared to other options but do not mention specific alternatives.
6. Some opinions might be long, it is ok to synthesize the most important information contained within them.
7. Some opinions might not contain relevant information for a product summary, it is ok to ignore them. For example: “After watching the show, I was looking forward to knowing more about the world, and I decided to give the book a chance.”

A.2 Quality Control

Throughout the process, we asked friends and colleagues to approve the quality of the summaries in the target languages (in their native language) before adding them to our train/test set. For all the languages, we could find enough Turkers to write summaries directly in the target languages given the selected sentences in English (although collecting Arabic was extremely slow). However, for Hindi since we were not getting quality summaries in a timely manner, we decided to collect summaries for two thirds of the test data through human translation of the English summaries instead of a combined summarization and translation process that we asked Turkers to do for other languages.

B LSA Hyper-parameters

For LSA, we considered the top 100 unigrams and bigrams appeared in less than 5% of sentences (to avoid most common terms) in the term-frequency matrix and used the top 5 eigenvalues for sorting the most informative sentences (i.e., focused on the 5 most important “topics” for each product). We then used the top 10 sentences as the sentences representing the reviews for each product. Table A1 shows an example of the top sentence in the reviews for a selected product.

C Fine-tuning Details

All the models are trained using Adam (Kingma and Ba, 2015) optimizer with $1e^{-5}$ learning rate, no warm-up steps, a linear learning rate scheduler, and an effective batch size of 112. For pre-fine-tuning, we trained the model for 2 epochs. But for fine-tuning, we trained the models for 5 epochs. For fine-tuning on FewSum data, we used 10 epochs (but as in the case of fine-tuning on CARS data, we included two shuffled versions of the input reviews during training).

D Sample Generated Summaries

A sample data point and generated summaries (not cherry picked) from the test set are provided in Table A1.

Product Name	Loud Alarm Clock with Bed Shaker, Vibrating Alarm Clock for Heavy Sleepers, Deaf and Hard of Hearing, Dual Alarm Clock, 2 Charger Ports, 7-Inch Display, Full Range Dimmer and Battery Backup - Green
Selected Sentences from Reviews	<p>"Outwardly they look great, the large green numbers give excellent readability and thanks to the adjustable brightness of the display, they Shine comfortably at night."</p> <p>The goods have been received Faster than I thought The workmanship is very fine Real time monitoring Ultrasonic alarm Very sensitive High precision A very satisfying shopping</p> <p>"I take sedatives and have slept through friends banging on my bedroom door/windows, fire trucks in my apartment parking lot right outside my door, and massive storms."</p> <p>"First off, I can sleep through a full blown tornado and over the years I've tried every trick in the book to wake me up in the mornings."</p> <p>"My sleep is quite heavy if I work hard during the day, I usually do not even hear my phone alarm or those loud mechanically ringing clocks."</p> <p>"Easy-to-see numbers with adjustable control, alarm sound, light weight, slim, occupying little space, it charges my mobile phone."</p> <p>"I put it on the bedside table, it is easy to read, USB plug to keep your mobile phone, notebook computer and any other you need to recharge."</p> <p>"Overall, I love this product and it's pretty good to wake up sleepy heads in the bed making it a great gift to buy another for my friend."</p> <p>"The extra large display allows even someone like me (-9 power eyesight) can see the time without glasses on, which is a very pleasant thing to experience."</p> <p>I decided on this clock because it had multiple alarm options and it had usb ports to charge my phone at bedside without taking up more plug space.</p>
Human Summary 1	The alarm clock has a loud ringer that will wake you up easily in the morning. The large green numbers on the clock are easy to read and it is easy to adjust the brightness of the display. The clock is small and takes up little space and there is even a USB port to charge devices.
Human Summary 2	The Loud Alarm Clock is designed to wake up heavy sleepers. It vibrates. It has two ports for charging, a seven-inch display, and a battery back up. Reviewers were pleased with the lightweight and slim design, the two chargers, the loud alarm sound and the large, easy-to-read display. There were no negative reviews.
Human Summary 3	This alarm clock is generally well liked by users. They mention that they love the connection ports and it does a good job waking them up.
Generated Summary English	The Loud Alarm Clock with Bed Shaker is a good product, easy to see with adjustable control, the numbers give excellent readability and the brightness of the display is adjustable. The product is lightweight and slim, occupying little space and it is easy to read. It has multiple alarm options and it has USB ports to charge your phone without taking up more plug space.
Generated Summary Spanish	El reloj de alarma ruidoso con sofocantedor de cama es una gran adición a su hogar. Es ideal para personas que tienen problemas de audición graves o no pueden oír el sonido de la alarma cuando duermen. Es fácil de leer y tiene una pantalla de 7 pulgadas que se puede utilizar para cargar sus teléfonos y portátiles. Los números verdes son fáciles de ver con un control ajustable y son ligeros y ligeros, lo que lo convierte en un gran regalo.
Generated Summary French	L'horloge d'alarme forte avec shaker de lit est un excellent ajout à votre équipement de nuit. Il est facile à lire et dispose d'un grand écran. Les chiffres verts sont faciles à lire. Ils s'allument confortablement la nuit et sont livrés avec une grande luminosité réglable. Il y a également deux ports USB pour charger votre téléphone portable et votre ordinateur portable.
Generated Summary Italian	L'orologio svegliare forte con Shaker Bed è un bel prodotto, facile da vedere con un controllo regolabile e un suono di allarme. Il prodotto è leggero e sottile, occupando poco spazio ed è facile da leggere. Il display è extra grande e permette anche a qualcuno di vedere l'ora senza occhiali accesi.

Table A1: An example of test data with the generated summaries in latin languages.

Augmenting Operations Research with Auto-Formulation of Optimization Models From Problem Descriptions

Rindra Ramamonjison^{1†}

Haley Li*

Timothy Yu[†]

Shiqi He*

Vishnu Rengan*

Amin Banitalebi-Dehkordi[†]

Zirui Zhou[†]

Yong Zhang[†]

Abstract

We describe an augmented intelligence system for simplifying and enhancing the modeling experience for operations research. Using this system, the user receives a suggested formulation of an optimization problem based on its description. To facilitate this process, we build an intuitive user interface system that enables the users to validate and edit the suggestions. We investigate controlled generation techniques to obtain an automatic suggestion of formulation. Then, we evaluate their effectiveness with a newly created dataset of linear programming problems drawn from various application domains. Code and data are available at <https://github.com/nl4opt/nl4opt-competition>

1 Introduction

Many real-world decision-making problems can be formulated and solved as mathematical optimization problems. The field of operations research (OR) has seen success in applications ranging from increasing bike-share ridership and efficiency (Beairsto et al., 2021; Ma et al., 2016), managing wastewater collection and treatment systems (Tao et al., 2020), to finding a revenue-maximizing pricing strategy (Bitran and Caldentey, 2016). In fact, optimization solvers can tackle different types of problems as they are powered by efficient algorithms such as the simplex method (Nash, 2000) or interior-point methods (Karmarkar, 1984).

However, modeling a problem into a proper formulation is a complex and time-consuming process. First, a domain expert must describe the problem and identify its variables, parameters, objective and constraints. Then, an OR expert needs to translate this problem description into a precise formulation using a modeling language, thus making the process inefficient and limits the accessibility of the solvers to non-experts (Hürlimann, 2013).

We propose an augmented intelligence system to simplify and enhance the modeling process. We partially automate the process using NLP models to suggest a formulation that the users can validate or edit the suggestions using an intuitive interface. Partial automation avoids the manual writing of the formulation by using a modeling language, thereby reducing the time and expertise to build optimization models. The intuitive interface also makes solvers more accessible to non-technical users.

From an NLP perspective, there are many challenges to parsing an optimization problem’s formulation from its natural language description:

- **Limited dataset.** The strenuous nature of modeling makes the cost of creating and labeling a large dataset prohibitive. Thus, efficient few-shot semantic parsers must be trained in a low-resource setting. Therefore, the solution must be built leveraging methods that excel on a limited training dataset.
- **Document-level input.** Most semantic parsers operate primarily at the sentence level. In contrast, long paragraph inputs describe the many variables, parameters, and constraints of optimization problems. Also, the parsing task involves a high level of compositionality and ambiguity.
- **Context-free output.** The outputs of most semantic parsing tasks share some contextual information with the inputs (e.g. database table or column names in SQL queries). In contrast, our task has a context-free tabular format, which makes it difficult to align the input-output pair.
- **Domain-agnostic parsing.** Finally, OR can tackle a diverse range of applications (Williams, 2013). Hence, the semantic parser must generalize well not only to new problem instances but also new application domains.

In this paper, we describe the underlying system that addresses these challenges and that enables

¹ rindranirina.ramamonjison@huawei.com

[†] Huawei Technologies Canada

* University of British Columbia, Vancouver

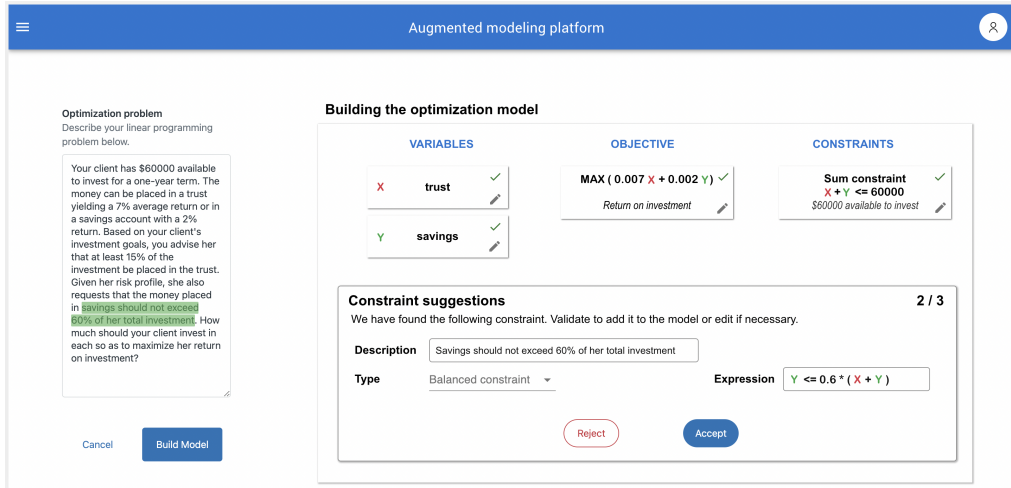


Figure 1: Augmented modeling platform.

the augmented modeling application. Our contributions are:

1. A novel augmented intelligence application that simplifies and enhances the modeling process.
2. New controllable generation methods for parsing the formulation of optimization problems from its natural language description.
3. The first dataset on linear programming word problems, with which we test and analyze the effectiveness of the methods for this emerging application.

2 Related Work

Augmented intelligence systems. Augmented intelligence systems use AI to assist (and not replace) the users in performing certain tasks. These systems could improve the experience or creativity of users in artistic applications such as story writing (Clark et al., 2018), music composition (Huang et al., 2020), poetry composition (Uthus et al., 2022), and sketching (Fan et al., 2019). This approach has also seen success in more task-oriented applications by helping teachers to grade homework efficiently (Malik et al., 2021), salespeople to summarize sales calls (Asi et al., 2022), and improving pneumonia diagnostic accuracy (Patel et al.). We adopt a similar approach for OR and focus on improving the modeling process.

Semantic parsing and generation. Semantic parsing maps natural language utterances into a machine-interpretable representation (Kamath and Das, 2019). This mapping has been extensively studied for output representations such as

SQL queries (Gan et al., 2020), Unix commands (Bharadwaj and Shevade, 2021), or logical forms for querying a knowledge base (Dong and Lapata, 2016). We tackle a different and challenging semantic parsing task as explained in Section 1.

Building on the success of attention models in sequence-to-sequence tasks (Sutskever et al., 2014; Luong et al., 2015), encoder-decoder architectures have been adopted for designing semantic parsers (Dong and Lapata, 2016, 2018; Wang et al., 2020). We propose using an intermediate representation (IR) that serves as a bridge between natural language and the canonical output format. Our two-stage mapping strategy is different from (Dong and Lapata, 2018), which initially generates a sketch of the query and then fills out the slots. In contrast to prior methods of constrained decoding (Hokamp and Liu, 2017; Scholak et al., 2021), our approach uses a simple beam search and leverages a prompt-guided generation and copying mechanism to guide the decoding.

Datasets on Mathematical World Problems (MWP). Recent works have studied the use of NLP models to automatically solve MWP (Wang et al., 2017; Ughade and Kumbhar, 2019). Most existing MWP datasets have focused on returning the solutions of elementary arithmetic problems (Roy and Roth, 2015; Koncel-Kedziorski et al., 2016) and algebra problems (Kushman et al., 2014; Huang et al., 2016). More challenging benchmarks have been recently proposed such as SVAMP (Patel et al., 2021), MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). In contrast, we build the first linear programming word problems (LPWP) dataset and evaluate methods of generating

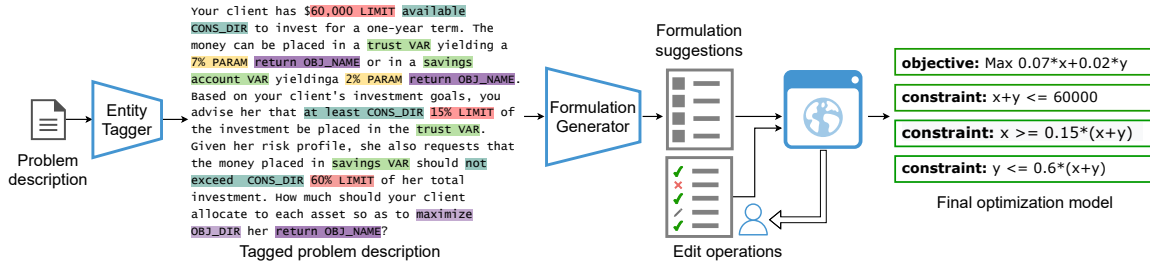


Figure 2: System diagram for augmented modeling framework.

the formulations as inputs to optimization solvers which can efficiently return an optimal solution.

3 Augmented Modeling Interface

We present an interactive system that enables users to model an optimization problem in collaboration with an AI system. To use this application, the user first describes the problem using natural language. Then, the system will suggest to the user the formulation of the optimization model including the decision variables, objective, and problem constraints. The system suggests the components of the formulation one at a time allowing the users to accept, reject, or edit the suggestions.

An example scenario of a portfolio optimization problem is shown in Figure 1. Here, the system suggested the formulation of a balance constraint given the description “*savings should not exceed 60% of her total investment*”. Had the system presented an incorrect formulation (e.g. an upper bound constraint), the user can change the type using the interface and the system reformulates the constraint expression automatically. The user can also edit the description, which will be stored as metadata of the model. In the same fashion, the user can add variables or constraints manually or edit the problem description to update the model. When the user is satisfied with the formulation, the system forwards it to an optimization solver, which then returns either an optimal solution of the problem to the user or warnings for some infeasible constraints.

Figure 2 shows an overview of the underlying auto-formulation system. It consists of an entity tagger, a formulation generator, and an augmented modeling interface. Given a problem description, the entity tagger labels the keywords that indicate the components of the optimization problem. For example, “return” and “at least” are tagged as an objective name and constraint direction, respectively. Then, the formulation generator uses the text description and the corresponding tagged entities to generate the formulation suggestions, which

are then presented to the user by the augmented modeling interface. In our implementation (experimental settings in the Appendix), we used an XLM-RoBERTa pre-trained transformer and fine-tuned it for entity recognition using the dataset described in Section 5.2.

4 OptGen: Controllable Generation of Optimization Formulation

Here, we present the methods behind our model OptGen, for generating the suggested formulation.

4.1 Two-stage mapping approach

It is difficult to directly map the problem description p to the formulation f due to the characteristics of the input-output pair (p, f) as mentioned in Section 1. First, the input document p can be unstructured and ambiguous especially when it describes many constraints of different types. From the input, we must precisely extract the canonical representation f of an optimization formulation.

As shown on the right of Figure 3, this canonical representation is a context-free table in which the column header is either a variable symbol or a constraint’s right-hand-side (rhs) limit. Each table row contains the parameters of the objective function or constraint. As a result, the canonical formulation f is context-free since it abstracts away the contextual information of p .

Instead, we adopt a two-stage mapping $p \mapsto r \mapsto f$ to bridge the gap between the natural language input and context-free formulation.

Text-to-IR mapping. We first define an intermediate representation (IR) r of the problem to simplify the parsing. As illustrated in Figure 3, a Text-to-IR mapping model generates a set of entity-typed declarations $\{D_i\}_{i=1}^n$ defined in an extended markup format to simplify its parsing. Note that other formats can be used for the IR (e.g. a format defined by a context-free grammar). Each declaration D_i is a sequence of tokens that represents

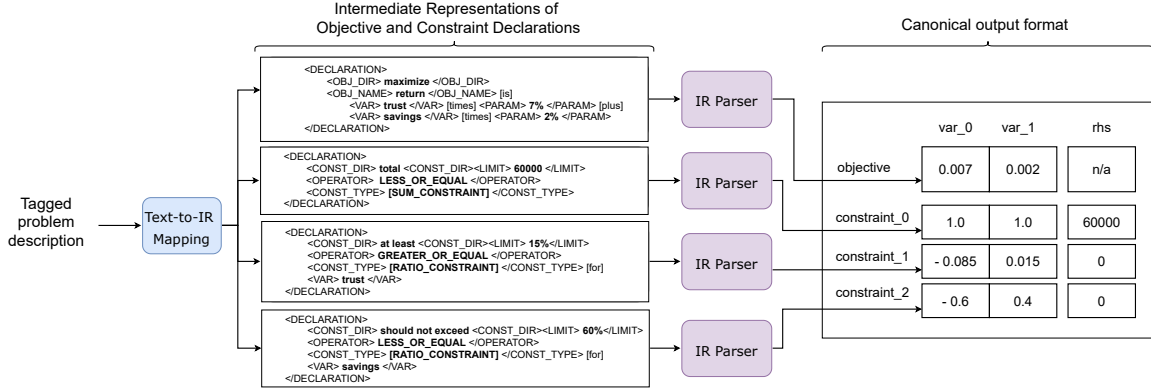


Figure 3: Overview of our formulation generation approach.

a typed and structured representation of either the optimization objective or a constraint. Each D_i is defined based on a predefined template of the different objective and constraint types. The tokens defining each declaration are wrapped in special tags. For example, `<OBJ_DIR> maximize </OBJ_DIR>` and `<OBJ_NAME> return </OBJ_NAME>` define the direction and the name of the objective. Since these tokens are derived from the input description, the IR preserves the context of the problem. We describe the grammar of the IR in the [Appendix](#).

IR parsing. An IR parser converts each IR declaration to the canonical format. We use an XML parser and apply simple transformations to convert numerical words to decimal numbers and to follow the following conventions. First, the canonical format always "minimize" a cost function. When the objective direction is "maximize," we instead change the sign of each objective parameter. Similarly, each inequality constraint must have a `LESS_OR_EQUAL` operator and convert each inequality constraint to the form $a^\top x \leq b$.

4.2 Prompt-guided generation model

We use an autoregressive model that is built upon the BART language model (Lewis et al., 2020). A prompt-guided generation is proposed to improve the accuracy of the Text-to-IR mapping. The idea is to decode the declarations of the IR one by one by using a declaration prompt to focus the generation. A declaration prompt is a prefix of the IR declaration of an objective or a constraint. For an objective, the prompt is composed of the entity tokens of the objective's direction and name. Similarly, the prompt for a constraint is defined by the entity tokens of the constraint direction. These tokens are obtained from the output of the Entity Tagger model shown in Figure 2. The declaration

prompt is added to the input text of the encoder. The role of the prompt is to provide contextual triggers for the decoder to focus on the relevant parts of the declaration to be generated. As illustrated in Figure 4, the model is trained to generate the IR of the declaration based on the declaration prompt and the problem description.

One key requirement of the Text-to-IR mapping is the ability to extract the variable names and data parameters from the descriptions and copy these important mentions from the input description into the output IR of the decoder. To augment the capability of BART encoder-decoder model, we leverage a copy mechanism that computes the probability distribution P_{copy} over the input tokens using cross-attention scores. The copy distribution is calculated at each time step t by taking the mean of the decoder's cross-attention scores across all attention heads as follows:

$$e_{t,i} = \frac{(W_s s_t)^T W_h h_i}{\sqrt{d_k}}$$

$$\alpha_{t,i} = \text{softmax}(e_{t,i})$$

$$P_{\text{copy}} = \frac{1}{n_H} \sum_i \alpha_{t,i}$$

where W_s and W_h are the projection matrices for the encoder and decoder. Then, s_t , h_i , and n_H are the decoder hidden state at time step t , the encoder hidden state for the attention head i , and the number of heads respectively.

We add the special tokens of the IR into the BART target vocabulary and mask out any vocabulary words that are not present in the source input. Following (See et al., 2017), we use a soft switch $p_{\text{gen}} \in [0, 1]$ to choose between generating a word from the vocabulary by sampling from P_{vocab} , or copying a word from the input sequence by sampling from P_{copy} . Thus, the final probability distri-

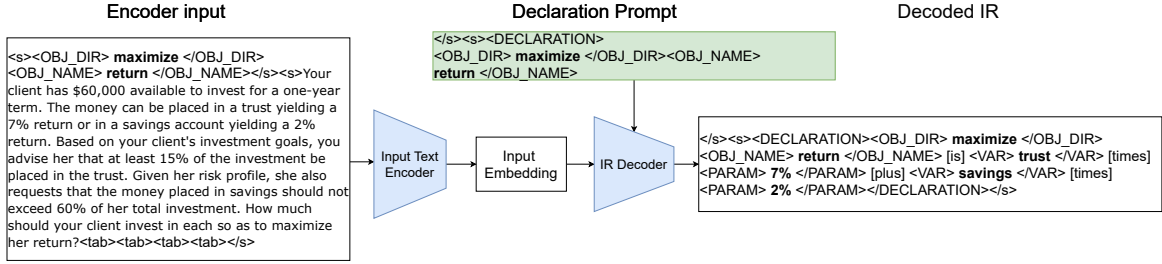


Figure 4: Description of prompt-guided generation method.

bution of a word w is given by:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) P_{\text{copy}}(w)$$

and is used to compute the loss for timestep t as the negative loglikelihood of the target word y_t for that timestep during training. Then, we average the loss over all time steps. As a result, the model is trained to produce tokens from either the IR vocabulary or the input problem description.

5 Experiments

5.1 Dataset

Dataset description. We curated a first-ever LPWP dataset¹ and use it to train and evaluate our methods. The dataset contains a collection of LP problems of the form:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{c}^\top \mathbf{x} \text{ s.t. } \mathbf{a}_i^\top \mathbf{x} \leq b_i, \quad i = 1, \dots, m$$

where \mathbf{c} and \mathbf{a}_i represent the parameters of the objective and the i -th constraint, respectively. b_i is the right-hand-side (rhs) limit, and the goal is to find \mathbf{x} that minimizes the objective value. The objective and constraint functions are linear with respect to variables in the LP problems. Each example has a text description of the problem and is annotated with the IR, math representation, and canonical formulation as shown in Figure 3. Table 1 summarizes the statistics of the dataset. More details and examples of the dataset are provided in the Appendix.

The dataset contains 1101 LP problems from the source domain (advertising, investment, sales) and target domain (production, science, transportation). The train, dev, and test splits contain 713, 99, and 289 samples, respectively. The training split is comprised of solely of samples from the source domain whereas the dev and test splits contain samples from both source and target domains with a source-to-target domain ratio of 1:3.

¹We plan to release this dataset and open-source the code to the research community for future research.

Number of Problems	1101
Number of Declarations	4216
Number of Constraint Types	6
Average Number of Variables	2.08
Average Number of Constraints	2.83

Table 1: Summary statistics of the LPWP dataset.

Dataset creation and quality control. The dataset was created from scratch and annotated internally by a team of 20 researchers and engineers with different levels of expertise in OR/NLP. The process included 3 stages: (1) problem creation, (2) NER annotation, and (3) REL annotation and declaration generation. Each stage was followed by rigorous quality checks and verification. The creation process is described in Figure 5.

Additional details of the dataset including the creation process, examples of the problem and their corresponding math formulation and IR, exclusion criteria, and inter-annotator agreement score, etc, are reported in the Appendix.

5.2 Results and Discussions

Baseline and metrics. We conducted experiments using the dataset described in Section 5.1. We use BART model (Lewis et al., 2020) as baseline for the two-stage mapping approach. In addition, we adopt the Text-to-Table (T2T) model (Wu et al., 2021) as a baseline for the direct approach that directly produces the canonical form. An example of its output is shown in Table 3 in the Appendix.

For evaluation, we measure the declaration-level mapping accuracy on the canonical formulation defined as:

$$\text{Acc} = 1 - \frac{\sum_{i=1}^N \min \{ \text{FP}_i + \text{FN}_i, D_i \}}{\sum_{i=1}^N D_i},$$

where for a given problem i , D_i is the number of ground-truth declarations, false positives FP_i is the number of non-matched predicted declarations, and false negatives FN_i is the number of excess unmatched ground-truth declarations. In other words,

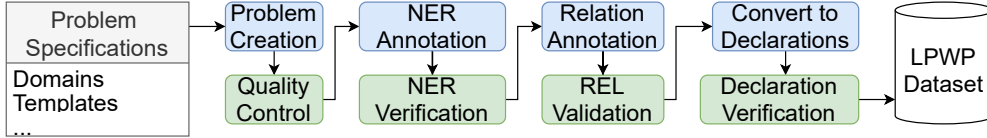


Figure 5: Overview of dataset creation.

FN_i is only non-zero when there are more ground-truth declarations than predicted declarations. The min is to prevent negative accuracy and overpenalization on single problems.

Main Results. Table 2 summarizes our results on the source and target domains. The BART baseline achieved the worst performance on all domains. In fact, BART hallucinated by producing too many constraints or many wrong parameters. Next, the direct mapping approach of T2T model achieved the highest accuracy of 88% on the Source domain but generalized poorly on out-of-domain (Target) test set. While it seemed to learn the task, T2T overfit to the Source training data. In contrast, our proposed OptGen achieved an absolute +18% accuracy improvement over direct mapping (T2T) on the Target domain. It was able to generalize better than the other models and produced the fewest errors on the problem structure and parameters. Our two-stage approach helped in this regard while the T2T must directly learn to convert to the canonical form. A detailed error analysis can be found in Figures 8 and 9 in the Appendix.

Ablation study. We also analyzed the importance of the individual methods used in our model in Table 2. It shows that the copy mechanism is important for accurately mapping the description to the equivalent formulation. Without the copy mechanism, we see significant accuracy drops of about 5% and 30% on Source and Target respectively. We show a qualitative comparison of generated IR formulations for two LP problems in Tables 12 and 13 in the Appendix. While our model could perfectly generate the correct representations, the model without copy mechanisms produced many errors. For example, it hallucinated the wrong constraint limits or detected the wrong constraint types. The prompt-guided generation method also led to slightly better performance on the Target domain. Finally, we noticed +6% improvement for T2T on the target domain when using contextual prompts.

These results show the importance of using controlled generation techniques for learning the syntax and grammar of the target IR language and for

Method	Source	Target	Sci	Prod	Trans
T2T	0.83	0.39	0.46	0.37	0.37
T2T + Prompt	0.88	0.45	0.49	0.46	0.40
BART	0.52	0.20	0.21	0.19	0.20
OptGen w/o copy	0.55	0.34	0.38	0.32	0.33
OptGen w/o prompt	0.58	0.61	0.64	0.63	0.57
OptGen	0.60	0.63	0.60	0.66	0.64

Table 2: Results for each model on the declaration-level mapping accuracy metric. Source consists of samples from the source domain test split. Target is a weighted mean (by number of declarations) of the science (Sci), production (Prod), and transportation (Trans) target domains test split.

accurately mapping the input description to the IR formulation.

Limitations and future works. Our preliminary results show the potential and the importance of controllable generation methods for enabling the augmented modeling system. While the proposed OptGen generation model was shown to generalize better than baseline models, its accuracy performance should still be enhanced further by improving the decoding method or by using edit-based models (Malmi et al., 2022) to automatically correct the erroneous parts of the formulation. As future work, we will conduct human evaluation of the augmented system by measuring the efficiency improvement perceived by real users. As the current dataset only covers LP problems, we will also expand it to cover other types of problems such as mixed-integer programs, which have different types of constraints. Other directions can also be explored to build more data-efficient methods.

6 Conclusion

We introduced an augmented modeling platform, in which users enter the descriptions of optimization problems and interact with an AI system to efficiently model their formulations. To this end, we described the underlying system for this emerging application and proposed controllable generation methods to enable it. We also created a training dataset of linear programming word problems to evaluate the effectiveness of the proposed methods. Our findings showed that the design of generation models and methods can have significant impact on the accuracy of the system’s suggested formula-

tions and that the system should help the users to validate and edit the suggestions.

7 Ethics Statement

This augmented intelligence system is intended to parse the formulation of optimization problems from its natural language description to aid stakeholders in their decision-making. The dataset was created taking special care to exclude samples with inappropriate language or names of real people, products or companies. The harm to users resulting from incorrect parsing is limited. However, depending on the application, the system may be used in sensitive or critical applications, such as a power grid, flights scheduling, etc. In such cases, the solver should be used with caution and the modeling process should be validated by the domain expert. Finally, operations research has historically been applied in tactical military operations. We must understand the potential negative impact of misusing this technology for society at large and the users must seriously consider the ethical concerns related to military applications.

References

- Abedelkadir Asi, Song Wang, Roy Eisenstadt, Dean Geckt, Yarin Kuper, Yi Mao, and Royi Ronen. 2022. [An end-to-end dialogue summarization system for sales calls](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 45–53, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.
- Jeneva Beairsto, Yufan Tian, Linyu Zheng, Qunshan Zhao, and Jinhyun Hong. 2021. [Identifying locations for new bike-sharing stations in glasgow: an analysis of spatial equity and demand factors](#). *Annals of GIS*, 0(0):1–16.
- Shikhar Bharadwaj and Shirish Shevade. 2021. [Explainable natural language to bash translation using abstract syntax tree](#). In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 258–267, Online. Association for Computational Linguistics.
- Gabriel R. Bitran and René A. Caldentey. 2016. An overview of pricing models for revenue management. *IEEE Engineering Management Review*, 44:134–134.
- Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. *23rd International Conference on Intelligent User Interfaces*.
- Karl Cobbe, Vineet Kosaraju, et al. 2021. [Training verifiers to solve math word problems](#). *arXiv preprint arXiv:2110.14168*.
- Li Dong and Mirella Lapata. 2016. [Language to logical form with neural attention](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.
- Li Dong and Mirella Lapata. 2018. [Coarse-to-fine decoding for neural semantic parsing](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.
- Judith E. Fan, Monica Dinculescu, and David Ha. 2019. [Collabdraw: An environment for collaborative sketching with an artificial agent](#). In *Proceedings of the 2019 on Creativity and Cognition*, page 556–561, New York, NY, USA. Association for Computing Machinery.
- Yujian Gan, Matthew Purver, and John R. Woodward. 2020. [A review of cross-domain text-to-SQL models](#). 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: Student Research Workshop*, pages 108–115, Suzhou, China. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.
- Chris Hokamp and Qun Liu. 2017. [Lexically constrained decoding for sequence generation using grid beam search](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.
- Cheng-Zhi Anna Huang, Hendrik Vincent Koops, Ed Newton-Rex, Monica Dinculescu, and Carrie J. Cai. 2020. [Ai song contest: Human-ai co-creation in songwriting](#).
- Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. [How well do computers solve math word problems? large-scale dataset construction and evaluation](#). In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany. Association for Computational Linguistics.
- Tony Hürlimann. 2013. *Mathematical modeling and optimization: an essay for the design of computer-based modeling tools*, volume 31. Springer Science & Business Media.

- Aishwarya Kamath and Rajarshi Das. 2019. [A survey on semantic parsing](#).
- N. Karmarkar. 1984. [A new polynomial-time algorithm for linear programming](#). In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, page 302–311, New York, NY, USA. Association for Computing Machinery.
- Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. [MAWPS: A math word problem repository](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.
- Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. [Learning to automatically solve algebra word problems](#). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.
- 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.
- 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.
- Yingying Ma, Xiaoran Qin, Jianmin Xu, and Xiangli Zou. 2016. [Research on pricing method of public bicycle service: A case study in guangzhou](#). In *IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 156–161.
- Ali Malik, Mike Wu, Vrinda Vasavada, Jinpeng Song, Madison Coots, John Mitchell, Noah D. Goodman, and Chris Piech. 2021. [Generative grading: Near human-level accuracy for automated feedback on richly structured problems](#). In *Proceedings of the 14th International Conference on Educational Data Mining, EDM 2021, virtual, June 29 - July 2, 2021*. International Educational Data Mining Society.
- Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. [Text generation with text-editing models](#).
- John C Nash. 2000. The (dantzig) simplex method for linear programming. *Computing in Science & Engineering*, 2(1):29–31.
- Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. [Are NLP models really able to solve simple math word problems?](#) In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.
- Bhavik N. Patel, Louis Rosenberg, Gregg Willcox, David Baltaxe, Mimi Lyons, Jeremy Irvin, Pranav Rajpurkar, Timothy Amrhein, Rajan Gupta, Safwan Halabi, Curtis Langlotz, Edward Lo, Joseph Mammarrappallil, A. J. Mariano, Geoffrey Riley, Jayne Seekins, Luyao Shen, Evan Zucker, and Matthew Lungren. [Human-machine partnership with artificial intelligence for chest radiograph diagnosis](#). *npj Digital Medicine*, 2(1).
- Subhro Roy and Dan Roth. 2015. [Solving general arithmetic word problems](#). In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. [Picard: Parsing incrementally for constrained auto-regressive decoding from language models](#).
- Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. [Get to the point: Summarization with pointer-generator networks](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. [Sequence to sequence learning with neural networks](#). In *Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS'14*, page 3104–3112, Cambridge, MA, USA.
- Diana Qing Tao, Martin Pleau, et al. 2020. [Analytics and Optimization Reduce Sewage Overflows to Protect Community Waterways in Kentucky](#). *Interfaces*, 50(1):7–20.
- Shounaak Ughade and Satish Kumbhar. 2019. [Survey on mathematical word problem solving using natural language processing](#). In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–5.
- David Uthus, Maria Voitovich, and R.j. Mical. 2022. [Augmenting poetry composition with Verse by Verse](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 18–26, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

- Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. [RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.
- Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. [Deep neural solver for math word problems](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.
- H Paul Williams. 2013. *Model building in mathematical programming*. John Wiley & Sons.
- Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. [Text-to-table: A new way of information extraction](#).

8 Appendix

This section contains supplementary materials including qualitative comparisons, examples from the dataset, as well as additional details which were omitted from the main paper due to space limitations.

A More details on dataset description

In Table 4, we showcase the different constraint types, along with their use in a problem, their math formulation, and their IR. Tables 6, 7 and 8 provide examples showing the problem description, IR and the math formulations.

B More details on dataset creation

The LPWP dataset creation process consists of three stages.

B.1 Stage 1: Problem creation

First, we invited 15 researchers and engineers to create original LP problems. They created the problems following some specifications such as problem domains and templates. The problem domains include advertising, investment, sales, agriculture, manufacturing, transportation and health sciences. During the creation step, 5 additional people were tasked with performing quality control and giving guidance to the problem creators. This included screening for the exclusion criteria (e.g. use of inappropriate language or names of real people, products or companies) and making sure the produced problem followed the specified domain and template. In fact, we used these templates to ensure the diversity of the problem structure (e.g. number of constraints, objectives and constraint types) in the dataset. If a mistake was detected in the problem, the problem creator would have been asked to fix it and the correction was verified. We made sure that each problem was verified by at least two people.

B.2 Stage 2: NER Annotation

At this stage, annotators were required to locate and classify entities mentioned in the problem. There are in total 6 types of targeted entities: variable (VAR), parameter (PARAM), limit (LIMIT), constraint direction (CONST_DIR), objective direction (OBJ_DIR) and objective name (OBJ_NAME). We used the Prodigy annotation tool to annotate the problems. A screenshot of one annotation session is provided in Figure 6. We asked an additional

annotator to resolve disagreements by reading the guideline thoughtfully and asking the annotators for clarifications if needed.

To ensure the quality of the NER annotations, four OR/NLP experts annotated more than 10% of the entire dataset, with an equal split between each domain, separately to compute the inter-annotator agreement. We measured an average pairwise micro-averaged F1 score of 97.7% for the inter-annotator agreement, showing the reliability of the annotation process.

B.3 Stage 3: Relation annotation and declaration generation

In the final stage, we annotated relations between entities for representing the objective and constraint declarations. Using a custom Prodigy annotation recipe, we integrated some validation checks to detect mistakes in the relation annotation period. These checks ensured that the annotator corrected all mistakes before proceeding to the next problem. Table 5 summarizes the relations required to represent each constraint type. The validation checks verified the number, labels, directions, and the entity labels of all relations that represent one constraint or objective. To increase the efficiency of the annotation process, we also provided guidelines and trained annotators to annotate each type of constraints. Similar to stage 2, we worked with annotators to resolve complex relations. A screenshot of a relation annotation session is provided in Figure 7.

After annotating the relations, we used a Python script to automatically convert the extracted relations into optimization declarations. A team of 5 people were asked to verify the correctness of the optimization declarations for all problems.

C Examples of target domain problems

Tables 9, 10 and 11 provide additional examples from the target domain: production, transportation, and sciences respectively.

D Predicted vs gold IR for target domain examples

Table 14-20 demonstrate the predictions of our model in comparison to the gold standard (ground-truth) for several examples from the target domain. We show examples for when the model produced the gold IR formulation in Table 14 and Table 15.

More importantly, we illustrate and analyze different errors when the model was used to parse examples from the target domain. For each example, detailed analysis of the errors are given in the captions of Table 16-20.

E Qualitative comparison of test predictions

In Table 12 and 13, we qualitatively compare the generated IR of two different optimization problems. Our model perfectly matched the gold IR for both problems. When the copy mechanism is not used, the model made few errors when generating the constraint declarations. In the first example (Table 12), the model hallucinates the wrong constraint limits in the first two constraints. Furthermore, the constraint type is invalid for the second constraint. Similar errors happen in the problem of Table 13.

F Experimental settings

NER experiments. We trained the XLM-RoBERTa transformer as the baseline model. The training was performed as a two-step approach with a training followed by a fine-tuning step. The training step utilized the HuggingFace `get_linear_schedule_with_warmup` function that uses a learning rate decreasing linearly to 0 from the initial learning rate (1E-4) after a warmup period. This step was run for a maximum of 25 epochs on a batch of 64 samples with the early stopping callback function set to stop training monitoring the loss of the development split with a patience of 5 epochs and minimum change of 0.001. A model checkpoint was also used to save a checkpoint of the model that performed the best on the development set. The fine-tuning step also utilized a learning rate of 0.0001 for a maximum of 30 epochs with the same callback functions used in the training step. To set the learning rate, we used a grid search using a development set.

Generation experiments. We trained the BART baseline model and our model for a total of 200 epochs, using a learning rate of 1E-06, and with a batch size of 32. The corresponding performance on the dev set for the best model is summarized in Table 21. We trained the Text-to-Table for a total of 4000 updates, using a learning rate of 1E-05, and with max-tokens of 4096. To set the learning rate, we used a grid search using a development set.

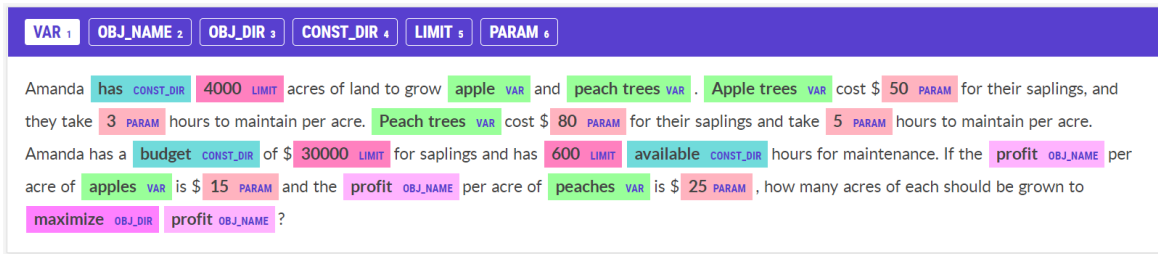


Figure 6: Screenshot of NER annotation using Prodigy.

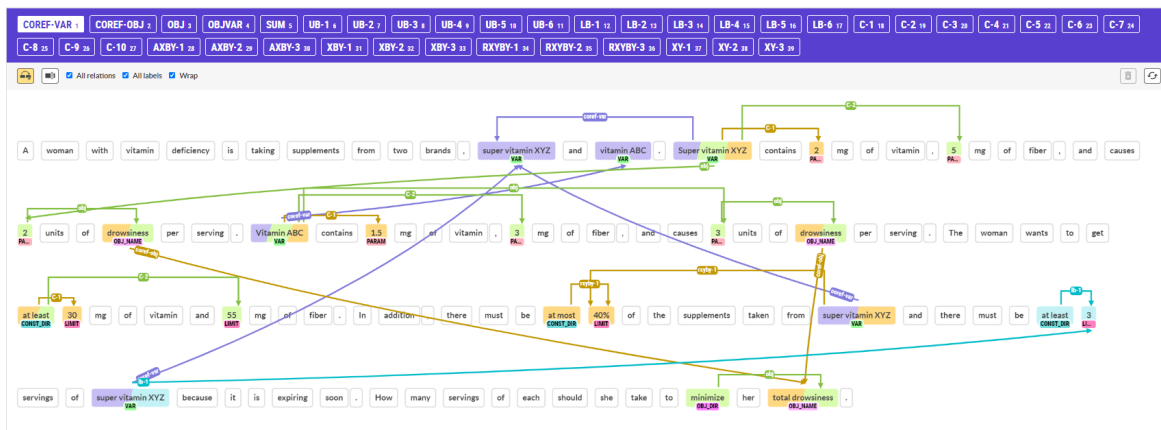


Figure 7: Screenshot of REL annotation using Prodigy.

Problem Description	A hotel employs cleaners and receptionists. Cleaners earn \$500 per week and receptionists earn \$350 per week. The hotel requires a minimum of 100 workers of whom at least 20 must be receptionists. To keep the hotel clean and running smoothly, the number of receptionists should be at least a third of the number of cleaners. The hotel wants to keep the weekly wage bill below \$30000. Formulate a LP to minimize the wage bill.			
Text-to-Table + Prompt Form		cleaners	receptionists	rhs
	objective	500.0	350.0	
	minimum	-1.0	-1.0	-100.0
	at least	0.0	-1.0	-20.0
	at least	0.3333	-1.0	0.0
	below	500.0	350.0	30000.0
Canonical Form		var_0	var_1	rhs
	objective	500.0	350.0	
	constraint_0	-1.0	-1.0	-100.0
	constraint_1	0.0	-1.0	-20.0
	constraint_2	0.3333	-1.0	0.0
	constraint_3	500.0	350.0	30000.0

Table 3: Text-to-Table example: Problem description, and expected output. Newline tokens have been replaced with newline characters, indentation was added, and column dividers were omitted for readability.

Constraint Type	Problem Description	Math Formulation + IR Representation
Sum Constraint	A bike shop sells two models of a bike: a mountain bike and a road bike. (...) The bike shop owner knows that the monthly demand will be at most 150 bikes . (...) How many bikes of each type should be stocked in order to maximize profit?	$x + y \leq 150$ <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 150 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE></DECLARATION>
Upper Bound	An ice cream bar sells vanilla and chocolate ice cream cones. (...) The ice cream bar must make at least 20 cones of vanilla ice cream but cannot make more than 50 cones . (...) How many cones of each flavor should they make to maximize profit?	$x \leq 50$ <DECLARATION><CONST_DIR> cannot make more than </CONST_DIR><LIMIT> 50 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> vanilla ice cream </VAR> </DECLARATION>
Lower Bound	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) A minimum of 10 adult doses need to be made. (...) How many of each dose should be prepared to maximize profit?	$y \geq 10$ <DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 10 </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> adult doses </VAR> </DECLARATION>
Linear Constraint (e.g. minimum requirement or capacity constraint)	A smoothie store sells two types of smoothies (...). Each small smoothie requires 2 units of ice cream and 1 unit of peanut butter. Each large smoothie requires 3 units of ice cream and 2 units of peanut butter. The company only has a total of 20 units of ice cream and 18 units of peanut butter. (...) how many of each should the store sell to maximize profit?	$2x + 3y \leq 20$ <DECLARATION><CONST_DIR> total </CONST_DIR><LIMIT> 20 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> small smoothie </VAR> [TIMES] <PARAM> 2 </PARAM><VAR> large smoothie </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION>
Ratio Control Constraint	A furniture store only stocks and sells dining tables and chairs. (...) Because chairs sell in larger quantities, at least 70% of all furniture in the store must be chairs . (...) Formulate an LP to maximize profit.	$x \geq 70/100 * (x+y)$ <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> chairs </VAR> </DECLARATION>
Balance Constraint Type-1 : $X \leq B*Y$	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) Demand is such that at least three times as many youth doses are needed than the adult doses . (...) . How many of each dose should be prepared to maximize profit?	$x \geq 3y$ <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> adult doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> youth doses </VAR></DECLARATION>
Balance Constraint Type-2 : $X \leq Y$	A peanut farmer has to send his product to the city. (...) He wants to spend at most \$3000 and the number of train trips must not exceed the number of truck trips . Formulate a LP to maximize the number of peanut packages that can be transported.	$x \leq y$ <DECLARATION><CONST_DIR> must not exceed </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [XY_CONSTRAINT] </CONST_TYPE><VAR> truck trips </VAR> [is] <VAR> train trips </VAR> </DECLARATION>

Table 4: Different types of constraints with examples from the LPWP dataset.

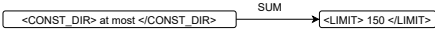
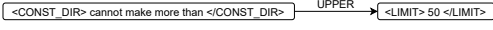


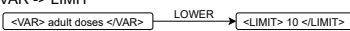



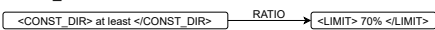

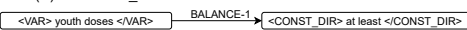


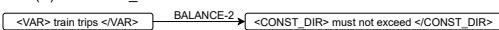

Constraint Type	Problem Description	Relation Annotation Examples
Sum Constraint	A bike shop sells two models of a bike: a mountain bike and a road bike. (...) The bike shop owner knows that the monthly demand will be at most 150 bikes . (...) How many bikes of each type should be stocked in order to maximize profit?	<ul style="list-style-type: none"> CONST_DIR -> LIMIT 
Upper Bound	An ice cream bar sells vanilla and chocolate ice cream cones. (...) The ice cream bar must make at least 20 cones of vanilla ice cream but cannot make more than 50 cones . (...) How many cones of each flavor should they make to maximize profit?	<ul style="list-style-type: none"> CONST_DIR -> LIMIT  VAR -> LIMIT 
Lower Bound	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) A minimum of 10 adult doses need to be made. (...) How many of each dose should be prepared to maximize profit?	<ul style="list-style-type: none"> CONST_DIR -> LIMIT  VAR -> LIMIT 
Linear Constraint (e.g. minimum requirement or capacity constraint)	A smoothie store sells two types of smoothies (...). Each small smoothie requires 2 units of ice cream and 1 unit of peanut butter. Each large smoothie requires 3 units of ice cream and 2 units of peanut butter. The company only has a total of 20 units of ice cream and 18 units of peanut butter. (...) how many of each should the store sell to maximize profit?	<ul style="list-style-type: none"> CONST_DIR -> LIMIT  VAR (X) -> PARAM  VAR (Y) -> PARAM 
Ratio Control Constraint	A furniture store only stocks and sells dining tables and chairs. (...) Because chairs sell in larger quantities, at least 70% of all furniture in the store must be chairs . (...) Formulate an LP to maximize profit.	<ul style="list-style-type: none"> CONST_DIR -> LIMIT  VAR -> LIMIT 
Balance Constraint Type-1 : X <= B*Y	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) Demand is such that at least three times as many youth doses are needed than the adult doses . (...) How many of each dose should be prepared to maximize profit?	<ul style="list-style-type: none"> VAR (X) -> CONST_DIR  CONST_DIR -> PARAM  PARAM -> VAR (Y) 
Balance Constraint Type-2 : X <= Y	A peanut farmer has to send his product to the city. (...) He wants to spend at most \$3000 and the number of train trips must not exceed the number of truck trips . Formulate a LP to maximize the number of peanut packages that can be transported.	<ul style="list-style-type: none"> VAR (X) -> CONST_DIR  CONST_DIR -> VAR (Y) 

Table 5: REL annotation examples for different constraint types.

Problem Description	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. Youth doses contain 20 grams of extract and adult doses contain 35 grams. Demand is such that at least three times as many youth doses are needed than the adult doses. A minimum of 10 adult doses need to be made. Youth doses are sold for a profit of \$5 while adult doses are sold at a profit of \$3. How many of each dose should be prepared to maximize profit?																				
Intermediate Representation	<pre> <DECLARATION> <OBJ_DIR> maximize </OBJ_DIR> <OBJ_NAME> profit </OBJ_NAME> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 5 </PARAM> <VAR> adult doses </VAR> [TIMES] <PARAM> 3 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> only </CONST_DIR><LIMIT> 5000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM> <VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM> </DECLARATION> DECLARATION> <CONST_DIR> at least </CONST_DIR> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE> <VAR> adult doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> youth doses </VAR> </DECLARATION> <DECLARATION> <CONST_DIR> minimum </CONST_DIR><LIMIT> 10 </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> adult doses </VAR> </DECLARATION> </pre>																				
Canonical Form	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>5</td> <td>3</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>20</td> <td>35</td> <td>5000</td> </tr> <tr> <td>constraint_1</td> <td>-1</td> <td>3</td> <td>0</td> </tr> <tr> <td>constraint_2</td> <td>0</td> <td>-1.0</td> <td>-10</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	5	3		constraint_0	20	35	5000	constraint_1	-1	3	0	constraint_2	0	-1.0	-10
	var_0	var_1	rhs																		
objective	5	3																			
constraint_0	20	35	5000																		
constraint_1	-1	3	0																		
constraint_2	0	-1.0	-10																		
Math Formulation	<pre> max 5x + 3y subject to 20x + 35y <= 5000 x >= 3y y >= 10 </pre>																				

Table 6: Original dataset - Resource allocation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	Your client has \$60,000 available to invest for a 1 year term. The money can be placed in a trust yielding a 2% return or in a savings account yielding a 3% return. Based on your experience, you advise your client that at least 15% of the investment be placed in the trust and that at most 80% of the investment be placed in the savings account. How much should your client invest in each so as to maximize his return on investment?																				
Intermediate Representation	<pre> <DECLARATION> <OBJ_DIR> maximize </OBJ_DIR> <OBJ_NAME> return </OBJ_NAME> [is] <VAR> trust </VAR> [TIMES] <PARAM> 2% </PARAM> <VAR> savings account </VAR> [TIMES] <PARAM> 3% </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 60,000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE> </DECLARATION> <DECLARATION> <CONST_DIR> at least </CONST_DIR><LIMIT> 15% </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> trust </VAR> </DECLARATION> <DECLARATION> <CONST_DIR> at most </CONST_DIR><LIMIT> 80% </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> savings account </VAR> </DECLARATION> </pre>																				
Canonical Form	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>0.02</td> <td>0.03</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>1</td> <td>1</td> <td>60000</td> </tr> <tr> <td>constraint_1</td> <td>-0.85</td> <td>0.15</td> <td>0</td> </tr> <tr> <td>constraint_2</td> <td>-0.8</td> <td>0.2</td> <td>0</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	0.02	0.03		constraint_0	1	1	60000	constraint_1	-0.85	0.15	0	constraint_2	-0.8	0.2	0
	var_0	var_1	rhs																		
objective	0.02	0.03																			
constraint_0	1	1	60000																		
constraint_1	-0.85	0.15	0																		
constraint_2	-0.8	0.2	0																		
Math Formulation	<pre> max (2/100)*x + (3/100)*y subject to x + y <= 60000 x >= (15/100)*(x+y) y <= (80/100)*(x+y) </pre>																				

Table 7: Original dataset - Investment allocation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	A farmer has 500 acres of land to grow turnips and pumpkins. Turnips require 50 minutes of watering and \$80 worth of pesticide per acre. Pumpkins require 90 minutes of watering and \$50 worth of pesticide per acre. The farmer has 40000 minutes available for watering and \$34000 available to spend on pesticide. If the revenue per acre of turnips is \$300 and the revenue per acre of pumpkins is \$450, how many acres of each should he grow to maximize his revenue.																				
Intermediate Representation	<pre> <DECLARATION> <OBJ_DIR> maximize </OBJ_DIR> <OBJ_NAME> revenue </OBJ_NAME> [is] <VAR> turnips </VAR> [TIMES] <PARAM> 300 </PARAM> <VAR> pumpkins </VAR> [TIMES] <PARAM> 450 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> has </CONST_DIR><LIMIT> 500 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 40000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Turnips </VAR> [TIMES] <PARAM> 50 </PARAM> <VAR> Pumpkins </VAR> [TIMES] <PARAM> 90 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 34000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Turnips </VAR> [TIMES] <PARAM> 80 </PARAM> <VAR> Pumpkins </VAR> [TIMES] <PARAM> 50 </PARAM> </DECLARATION> </pre>																				
Canonical Form	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>300</td> <td>450</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>1</td> <td>1</td> <td>500</td> </tr> <tr> <td>constraint_1</td> <td>50</td> <td>90</td> <td>40000</td> </tr> <tr> <td>constraint_2</td> <td>80</td> <td>50</td> <td>34000</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	300	450		constraint_0	1	1	500	constraint_1	50	90	40000	constraint_2	80	50	34000
	var_0	var_1	rhs																		
objective	300	450																			
constraint_0	1	1	500																		
constraint_1	50	90	40000																		
constraint_2	80	50	34000																		
Math Formulation	<pre> max 300x + 450y subject to x + y <= 500 50x + 90y <= 40000 80x + 50y <= 34000 </pre>																				

Table 8: Original Dataset - Farming example: problem description, intermediate representation, canonical form, and math formulation.

<p>Problem Description</p>	<p>A mining company has available a total of 100 square miles of mining sites and considering the use of two mining techniques: heap leaching and vat leaching. For each square mile of land, heap leaching technique can have a daily production of 3 tons of rare earth oxide per square miles but it also creates 8 tons of polluted wastewater and requires 10 extraction machines. On the other hand, vat leaching technique produces 5 tons of rare earth oxide per square miles per day while creating 17 tons of polluted wastewater and requiring 20 extraction machines. There are 100 machines available and due to environmental regulations, the amount of polluted wastewater must be at most 90 tons daily. Find the proportion of lands that use each mining technique in order to maximize the daily production of rare earth oxide.</p>																				
<p>Intermediate Representation</p>	<pre> <DECLARATION> <OBJ_DIR> maximize </OBJ_DIR> <OBJ_NAME> rare earth oxide </OBJ_NAME> [is] <VAR> heap leaching </VAR> [TIMES] <PARAM> 3 </PARAM> <VAR> vat leaching </VAR> [TIMES] <PARAM> 5 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 100 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> heap leaching </VAR> [TIMES] <PARAM> 10 </PARAM> <VAR> vat leaching </VAR> [TIMES] <PARAM> 20 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> at most </CONST_DIR><LIMIT> 90 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> heap leaching </VAR> [TIMES] <PARAM> 8 </PARAM> <VAR> vat leaching </VAR> [TIMES] <PARAM> 17 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 100 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE> </DECLARATION> </pre>																				
<p>Canonical Form</p>	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>3</td> <td>5</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>10</td> <td>20</td> <td>100</td> </tr> <tr> <td>constraint_1</td> <td>8</td> <td>17</td> <td>90</td> </tr> <tr> <td>constraint_2</td> <td>1</td> <td>1</td> <td>100</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	3	5		constraint_0	10	20	100	constraint_1	8	17	90	constraint_2	1	1	100
	var_0	var_1	rhs																		
objective	3	5																			
constraint_0	10	20	100																		
constraint_1	8	17	90																		
constraint_2	1	1	100																		

Math Formulation	<pre>max 3x + 5y subject to 10x + 20y <= 100 8x + 17y <= 90 x + y <= 100</pre>
-------------------------	---

Table 9: Out-of-domain dataset - Production example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	<p>A shipping company need to transport packages by either truck or car. A truck can transport 50 packages per trip while a car can transport 30 packages per trip. In addition, a truck uses 20 liters of gas per trip while a car uses 15 liters of gas per trip. There can be at most 5 truck trips made and at least 30% of all the trips must be made by car. The company needs to transport at least 500 packages. How many of each transportation should they use to minimize the total amount of gas consumed?</p>																				
Intermediate Representation	<pre> <DECLARATION> <OBJ_DIR> minimize </OBJ_DIR> <OBJ_NAME> amount of gas </OBJ_NAME> [is] <VAR> truck </VAR> [TIMES] <PARAM> 20 </PARAM> <VAR> car </VAR> [TIMES] <PARAM> 15 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> at most </CONST_DIR><LIMIT> 5 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> truck </VAR> </DECLARATION> <DECLARATION> <CONST_DIR> at least </CONST_DIR><LIMIT> 30% </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> car </VAR> </DECLARATION> <DECLARATION> <CONST_DIR> at least </CONST_DIR><LIMIT> 500 </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM> <VAR> car </VAR> [TIMES] <PARAM> 30 </PARAM> </DECLARATION> </pre>																				
Canonical Form	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>20</td> <td>15</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>1</td> <td>0</td> <td>5</td> </tr> <tr> <td>constraint_1</td> <td>0.3</td> <td>-0.7</td> <td>0</td> </tr> <tr> <td>constraint_2</td> <td>-50</td> <td>-30</td> <td>-500</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	20	15		constraint_0	1	0	5	constraint_1	0.3	-0.7	0	constraint_2	-50	-30	-500
	var_0	var_1	rhs																		
objective	20	15																			
constraint_0	1	0	5																		
constraint_1	0.3	-0.7	0																		
constraint_2	-50	-30	-500																		
Math Formulation	<pre> min 20x + 15y subject to x <= 5 y >= (30/100)*(x+y) 50x + 30y >= 500 </pre>																				

Table 10: Out-of-domain dataset - Transportation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	<p>A patient is undergoing radiation treatment involving two beams, Beam 1 and Beam 2. Beam 1 delivers a dose of 0.3 units of medicine per minute to the benign area of the pancreas and 0.2 units of medicine per minute to the benign area of the skin. Beam 2 delivers 0.2 units of medicine per minute to the benign area of the pancreas and 0.1 units of medicine per minute to the benign area of the skin. In addition, beam 1 delivers 0.6 units of medicine per minute to the tumor and beam 2 delivers 0.4 units of medicine per minute to the tumor. At most 4 units of medicine should be received by the skin and at least 3 units of medicine should be delivered to the tumor. How many minutes of each beam should be used to minimize the total radiation received by the pancreas?</p>																
Intermediate Representation	<pre> <DECLARATION> <OBJ_DIR> minimize </OBJ_DIR> <OBJ_NAME> total radiation received by the pancreas </OBJ_NAME> [is] <VAR> Beam 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM> <VAR> Beam 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> At most </CONST_DIR><LIMIT> 4 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Beam 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM> <VAR> Beam 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> beam 1 </VAR> [TIMES] <PARAM> 0.6 </PARAM> <VAR> beam 2 </VAR> [TIMES] <PARAM> 0.4 </PARAM> </DECLARATION> </pre>																
Canonical Form	<table border="1"> <thead> <tr> <th></th> <th>var_0</th> <th>var_1</th> <th>rhs</th> </tr> </thead> <tbody> <tr> <td>objective</td> <td>0.3</td> <td>0.2</td> <td></td> </tr> <tr> <td>constraint_0</td> <td>0.2</td> <td>0.1</td> <td>4</td> </tr> <tr> <td>constraint_1</td> <td>-0.6</td> <td>-0.4</td> <td>3</td> </tr> </tbody> </table>		var_0	var_1	rhs	objective	0.3	0.2		constraint_0	0.2	0.1	4	constraint_1	-0.6	-0.4	3
	var_0	var_1	rhs														
objective	0.3	0.2															
constraint_0	0.2	0.1	4														
constraint_1	-0.6	-0.4	3														
Math Formulation	<pre> min 0.3x + 0.2y subject to 0.2x + 0.1y <= 4 0.6x + 0.4y >= 3 </pre>																

Table 11: Out-of-domain dataset - Health Science example: problem description, intermediate representation, canonical form, and math formulation.

A furniture store only stocks and sells dining tables and chairs. The profit per dining table is \$350 and the profit per chair is \$75. There is 500 sq ft of space available and a dining table requires 8 sq ft of floor space while a chair requires 2 sq ft. Because chairs sell in larger quantities, at least 70% of all furniture in the store must be chairs. In terms of capital, a dining table ties up \$1000 in capital and a chair ties up \$150 in capital. The company wants a maximum of \$20000 worth of capital tied up at any time. Formulate an LP to maximize profit.

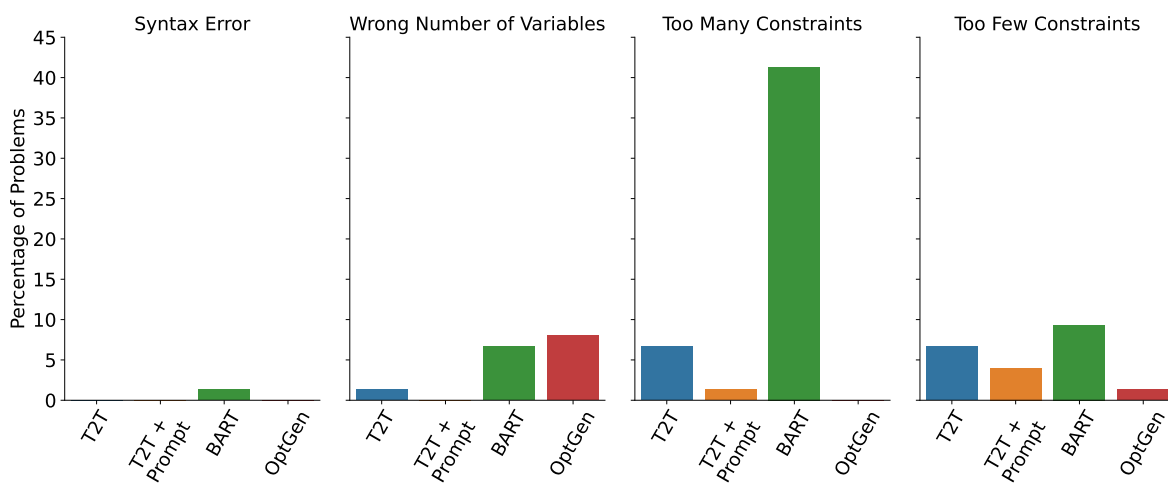
OptGen w/o copy	OptGen
<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> profit </OBJ_NAME> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 350 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 75 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> available </CONST_DIR><LIMIT> 20000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 8 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 2 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM> </DECLARATION> <DECLARATION><CONST_DIR> maximum </CONST_DIR><LIMIT> 20000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> profit </OBJ_NAME> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 350 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 75 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> available </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 8 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 2 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> chairs </VAR> </DECLARATION> <DECLARATION><CONST_DIR> maximum </CONST_DIR><LIMIT> 20000 </LIMIT> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM></DECLARATION> </pre>

Table 12: Qualitative comparison of model predictions with and without copy mechanism. In this example, the model without copy mechanism produced erroneous IR declarations (highlighted in red) whereas our model produces perfect matches with the gold declarations.

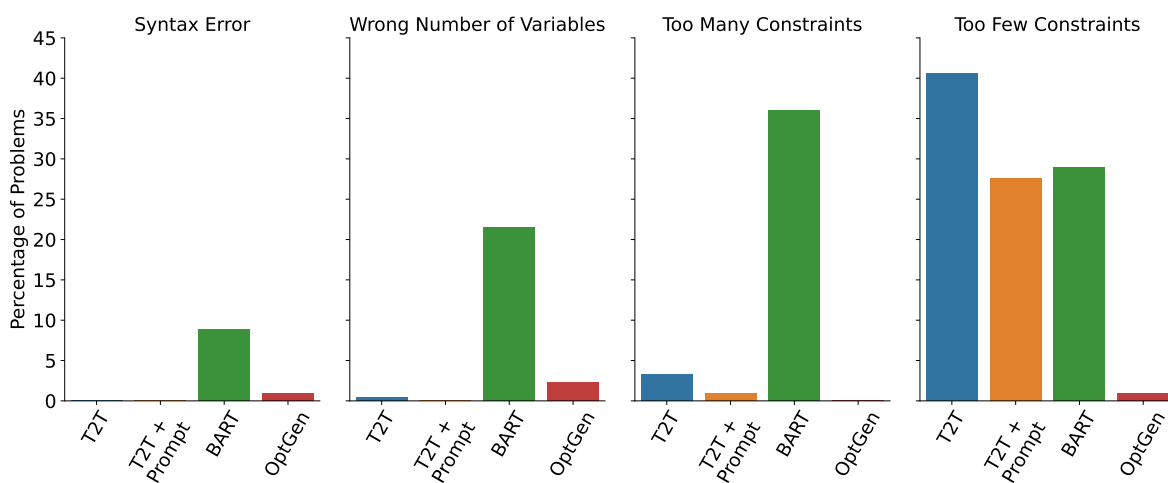
There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. Youth doses contain 20 grams of extract and adult doses contain 35 grams. Demand is such that at least three times as many youth doses are needed than the adult doses. A minimum of 10 adult doses need to be made. Youth doses are sold for a profit of \$5 while adult doses are sold at a profit of \$3. How many of each dose should be prepared to maximize profit?

OptGen w/o copy	OptGen
<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> profit </OBJ_NAME> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> only </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> youth doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> adult doses </VAR></DECLARATION> <DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> profit </OBJ_NAME> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> only </CONST_DIR><LIMIT> 5000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> adult doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> youth doses </VAR></DECLARATION> <DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> adult doses </VAR></DECLARATION> </pre>

Table 13: Qualitative comparison of model predictions with and without copy mechanism. In this example, the model without copy mechanism produced erroneous IR declarations (highlighted in red) whereas our model produces perfect matches with the gold declarations.

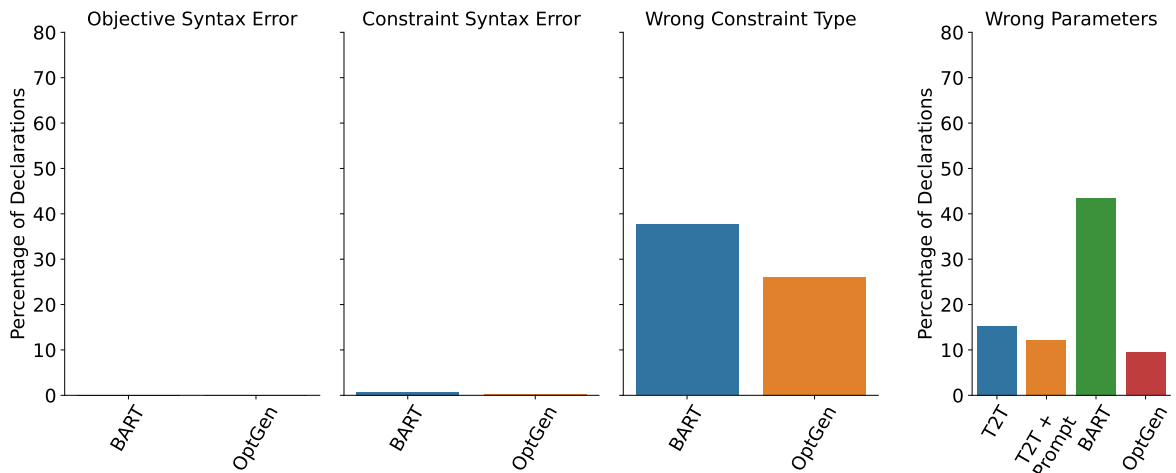


(a) Problem level errors on the Source domain

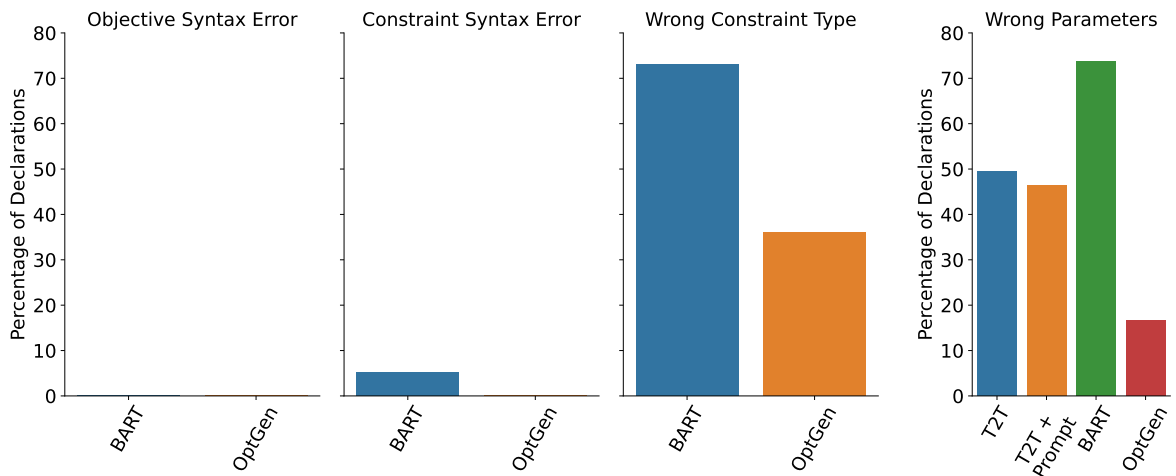


(b) Problem level errors on the Target domain

Figure 8: Classification of problem level errors for each model. These errors are not mutually exclusive. Problem level syntax errors will result in a completely incorrect problem as they cannot be parsed. On models predicting the IR of a problem, a problem level syntax error is defined as one that cannot be parsed through an XML parser. On the T2T models, syntax errors are defined as a mismatch between the number of columns in each row.



(a) Declaration level errors on the Source domain.



(b) Declaration level errors on the Target domain.

Figure 9: Classification of declaration level errors for each model. Note that some of these error types are not made by the T2T models, as they do not predict constraint types, and their syntax errors are all at the problem level. The parser will skip parsing poorly formatted declarations, which are later counted as incorrect. We define a declaration level parameter error as a declaration that contains any parameter mismatch between the prediction and the ground truth.

A patient in the hospital can take two pills, Pill 1 and Pill 2. Per pill, pill 1 provides 0.2 units of pain medication and 0.3 units of anxiety medication. Per pill, pill 2 provides 0.6 units of pain medication and 0.2 units of anxiety medication. In addition, pill 1 causes 0.3 units of discharge while pill 2 causes 0.1 units of discharge. At most 6 units of pain medication can be provided and at least 3 units of anxiety medication must be provided. How many pills of each should the patient be given to minimize the total amount of discharge?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of discharge </OBJ_NAME> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> At most </CONST_DIR><LIMIT> 6 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.6 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of discharge </OBJ_NAME> [is] <VAR> Pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> At most </CONST_DIR><LIMIT> 6 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.6 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM></DECLARATION> </pre>

Table 14: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, the predicted IR is almost equal to the gold IR except for the extra token ("total") in the objective name declaration.

An international goods exporter uses ships and planes to transport goods. A ship can take 40 containers worth of goods and uses 500 liters of fuel per trip. A plane can take 20 containers worth of goods and uses 300 liters of fuel per trip. The company needs to transport at least 500 containers worth of goods. In addition, there can be at most 10 plane trips made and a minimum of 50% of the trips made must be by ship. How many of each trip should be made to minimize the total amount of fuel consumed?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of fuel </OBJ_NAME> [is] <VAR> ship </VAR> [TIMES] <PARAM> 500 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 300 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> ship </VAR> [TIMES] <PARAM> 40 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 20 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><LIMIT> 10 </LIMIT><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> plane </VAR></DECLARATION> <DECLARATION><CONST_DIR> minimum </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 50% </LIMIT><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> ship </VAR></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of fuel </OBJ_NAME> [is] <VAR> ship </VAR> [TIMES] <PARAM> 500 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 300 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> ship </VAR> [TIMES] <PARAM> 40 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 20 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><LIMIT> 10 </LIMIT><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> plane </VAR></DECLARATION> <DECLARATION><CONST_DIR> minimum </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 50% </LIMIT><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> ship </VAR></DECLARATION> </pre>

Table 15: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model perfectly matched the gold IR.

A pharmacy has 3000 mg of morphine to make painkillers and sleeping pills. Each painkiller pill requires 10 mg of morphine and 3 units of digestive medicine. Each sleeping pill requires 6 mg of morphine and 5 units of digestive medicine. The pharmacy needs to make at least 50 painkiller pills. Since sleeping pills are more popular, at least 70% of the pills should be sleeping pills. How many of each should the pharmacy make to minimize the total amount of digestive medicine needed?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of digestive medicine </OBJ_NAME> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 3 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> sleeping pills </VAR></DECLARATION> <DECLARATION><CONST_DIR> has </CONST_DIR><LIMIT> 3000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 10 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of digestive medicine </OBJ_NAME> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 3 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION> <DECLARATION><CONST_DIR> has </CONST_DIR><LIMIT> 3000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 10 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION> </pre>

Table 16: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, our model predicted the wrong constraint type as lower bound instead of a ratio constraint. This shows that it is sometimes hard for the model to distinguish between similar constraint types. The rest of the declaration that follows the constraint type is also invalid.

A parent feeds their baby two flavors of baby food, apple and carrot, in order to meet the babies fat and folate requirements. Each serving of apple flavored baby food contains 2 units of fat and 5 units of folate. Each serving of carrot flavored baby food contains 4 units of fat and 3 units of folate. The baby does not like the carrot flavor, and therefore he must eat three times as many apple flavored baby food as carrot flavored baby food. However, he must eat at least 2 servings of carrot flavored baby food. If the baby can consume at most 100 units of folate, how many servings of each should he eat to maximize his fat intake?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> fat intake </OBJ_NAME> [is] <VAR> apple flavored baby </VAR> [TIMES] <PARAM> 2 </PARAM> <VAR> carrot flavored baby </VAR> [TIMES] <PARAM> 4 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> must eat </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> carrot flavored baby </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> apple flavored baby food </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 2 </LIMIT><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> carrot flavored baby </VAR></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 100 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> apple flavored baby </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> carrot flavored baby </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> fat intake </OBJ_NAME> [is] <VAR> apple flavored baby food </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> must eat </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> carrot flavored baby food </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> apple flavored baby food </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 2 </LIMIT><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> carrot flavored baby food </VAR></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 100 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> apple flavored baby food </VAR> [TIMES] <PARAM> 2 </PARAM><VAR> carrot flavored baby food </VAR> [TIMES] <PARAM> 4 </PARAM></DECLARATION> </pre>

Table 17: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, the model produced an erroneous declaration in the objective and the last constraint. The wrong data parameters, which should describe the objective function, are instead copied into the last constraint. This example illustrates the difficulty of parsing an input document that is ambiguous.

A chocolate company can transport their boxes of chocolate either using their own vans or by renting trucks. Their vans can transport 50 boxes per trip while a truck can transport 80 boxes per trip. Since they own their vans, the cost per van trip is \$30 while the cost per truck trip is \$50. The company needs to transport at least 1500 boxes of chocolate and they have a budget of \$1000. Since the vans also provide advertising, the number of trips by van must be larger than the number of trips by trucks. How many of trip by each should be done to minimize the total number of trips?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> number of trips </OBJ_NAME> [is] <VAR> vans </VAR> [TIMES] <PARAM> ONE </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> ONE </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 80 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> budget </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> van </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> must be larger than </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XY_CONSTRAINT] </CONST_TYPE> <VAR> trucks </VAR> [is] <VAR> van </VAR> *) </DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total number of trips </OBJ_NAME> [is] <VAR> van </VAR> [TIMES] <PARAM> ONE </PARAM><VAR> truck </VAR> [TIMES] <PARAM> ONE </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> 80 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> budget </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> van </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> must be larger than </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> 80 </PARAM> </DECLARATION> </pre>

Table 18: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, our model detects a linear constraint instead of a balance constraint. In fact, the balance constraints are less frequent in the training dataset whereas the linear constraints are the majority ones. This can explain this type of error as the constraint types are imbalanced in the training dataset.

A toy store decides to deliver gifts using two shipping companies, a new one and an old one. The new company can deliver 50 gifts per trip while the old company can deliver 70 gifts per trip. The new company uses 30 liters of diesel per trip while the old company uses 40 liters of diesel per trip. The toy store needs to deliver at least 1000 gifts. There can be at most 15 trips made by the new company. In order to make sure that the old company does not go out of business, at least 40% of all trips must be made by the old company. How many trips should each company make to minimize the total amount of diesel used?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of diesel </OBJ_NAME> [is] <VAR> new company </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 40 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 15 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> new company </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 40% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> old company </VAR></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of diesel </OBJ_NAME> [is] <VAR> new company </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 40 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 15 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION> </pre>

Table 19: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model incorrectly generated the expressions for the last two constraints. In fact, it detects the wrong constraint types and produced the same invalid algebraic expression. This suggests that the generation could be made more precise by adding additional constrain context into the declaration prompt to distinguish between different constraints.

A zoo needs to transport their monkeys to the vet either by bus or by car. A bus can transport 20 monkeys per trip and takes 30 minutes. A car can transport 6 monkeys per trip and takes 15 minutes. There can be at most 10 bus trips. In addition, since the monkeys get aggressive when there are too many in one place at least 60% of the trips should be by car. If the zoo needs to transport 300 monkeys, how many trips of each should be done to minimize the total time required to transport the monkeys?

Gold	Pred
<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total time </OBJ_NAME> [is] <VAR> bus </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 15 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> bus </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 60% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> car </VAR></DECLARATION> <DECLARATION><CONST_DIR> needs </CONST_DIR><LIMIT> 300 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> bus </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION> </pre>	<pre> <DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total time </OBJ_NAME> [is] <VAR> monkeys </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> cars </VAR> [TIMES] <PARAM> 15 </PARAM></DECLARATION> <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> bus </VAR></DECLARATION> <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 60% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> car </VAR></DECLARATION> <DECLARATION><CONST_DIR> needs </CONST_DIR><LIMIT> 300 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> bus </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION> </pre>

Table 20: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model detects an invalid decision variable "monkeys" in the predicted objective declaration.

Method	Accuracy	Rouge-1			Rouge-2			Rouge-L		
		Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
OptGen	0.61	0.89	0.89	0.89	0.80	0.80	0.79	0.86	0.86	0.86

Table 21: Performance of our model on development set. Rouge scores included declaration tags to illustrate how well the model is able to reproduce syntax.

Knowledge Distillation based Contextual Relevance Matching for E-commerce Product Search

Ziyang Liu[§], Chaokun Wang^{§*}, Hao Feng[§], Lingfei Wu[†], Liqun Yang[‡]

[§]Tsinghua University, [†]JD.com, [‡]CNAEIT

[§]liu-zy21@mails.thu.edu.cn, [§]chaokun@thu.edu.cn

[§]fh20@mails.thu.edu.cn, [†]lwu@email.wm.edu, [‡]yanglq@cnaeit.com

Abstract

Online relevance matching is an essential task of e-commerce product search to boost the utility of search engines and ensure a smooth user experience. Previous work adopts either classical relevance matching models or Transformer-style models to address it. However, they ignore the inherent bipartite graph structures that are ubiquitous in e-commerce product search logs and are too inefficient to deploy online. In this paper, we design an efficient knowledge distillation framework for e-commerce relevance matching to integrate the respective advantages of Transformer-style models and classical relevance matching models. Especially for the core student model of the framework, we propose a novel method using k -order relevance modeling. The experimental results on large-scale real-world data (the size is 6~174 million) show that the proposed method significantly improves the prediction accuracy in terms of human relevance judgment. We deploy our method to JD.com online search platform. The A/B testing results show that our method significantly improves most business metrics under price sort mode and default sort mode.

1 Introduction

Relevance matching (Guo et al., 2016; Rao et al., 2019; Wang et al., 2020) is an important task in the field of ad-hoc information retrieval (Zhai and Lafferty, 2017), which aims to return a sequence of information resources related to a user query (Huang et al., 2020; Chang et al., 2021; Sun and Duh, 2020). Generally, texts are the dominant form of user queries and returned information resources. Given two sentences, the target of relevance matching is to estimate their relevance score and then judge whether they are relevant or not. However, text similarity does not mean semantic similarity. For example, while “mac pro 1.7GHz” and “mac

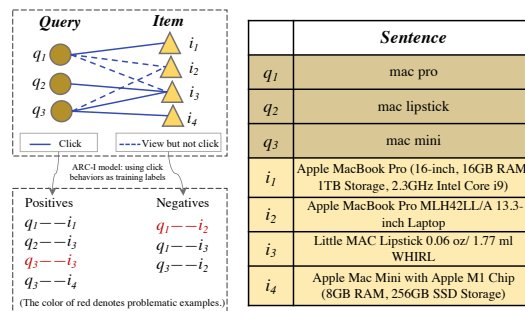


Figure 1: Shortcoming of the existing relevance matching model. Here we take the ARC-I model as an example. The right part shows the ground truth of queries and item titles. The left part shows two problematic examples in ARC-I, which deviate from the ground truth.

lipstick 1.7ml” look alike, they describe two different and irrelevant products. Therefore, relevance matching is important, especially for long-term user satisfaction of e-commerce search (Niu et al., 2020; Xu et al., 2021; Zhu et al., 2020).

Recently, Transformer-style models (e.g., BERT (Devlin et al., 2019) and ERNIE (Sun et al., 2019b)) have achieved breakthroughs on many NLP tasks and shown satisfactory performance on relevance matching, but they are hard to deploy to the online environment due to their high time complexity. Moreover, these methods cannot deal with the abundant context information (i.e., the neighbor features in a query-item bipartite graph) in e-commerce product search. Last but not least, when applied to real-world scenarios, existing classical relevance matching models directly use user behaviors as labeling information (Figure 1). However, this solution is not directly suitable for relevance matching because user behaviors are often noisy and deviate from relevance signals (Mao et al., 2019; Liu and Mao, 2020).

In this paper, we propose to incorporate bipartite graph embedding into the knowledge distillation framework (Li et al., 2021; Dong et al., 2021; Rashid et al., 2021; Wu et al., 2021b; Zhang et al.,

*Chaokun Wang is the corresponding author.

2020) to solve the relevance matching problem in the scene of e-commerce product search. We adopt BERT (Devlin et al., 2019) as the teacher model in this framework. Also, we design a novel model called BERM, **B**ipartite graph **E**mbedding for **R**elevance **M**atching (BERM), which acts as the student model in our knowledge distillation framework. This model captures the 0-order relevance using a word interaction matrix attached with positional encoding and captures the higher-order relevance using the metapath embedding with graph attention scores. For online deployment, it is further distilled into a tiny model BERM-O.

Our main contributions are as follows:

- We formalize the k -order relevance problem in a bipartite graph (Section 2.1) and address it by a knowledge distillation framework with a novel student model called BERM.
- We apply BERM to the e-commerce product search scene with abundant context information (Section 2.4) and evaluate its performance (Section 3). The results indicate that BERM outperforms the state-of-the-art methods.
- To facilitate online applications, we further distill BERM into a faster model, i.e., BERM-O. The results of online A/B testing indicate that BERM-O significantly improves most business metrics under price sort mode and default sort mode.

2 Method

2.1 Problem Definition

We first give the definition of the bipartite graph:

Definition 1 Bipartite Graph. Given a graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, it contains two disjoint node sets $\mathcal{U} : \{u_1, u_2, \dots, u_n\}$ and $\mathcal{V} : \{v_1, v_2, \dots, v_{n'}\}$. For edge set $\mathcal{E} : \{e_1, \dots, e_m\}$, each edge e_i connects u_j in \mathcal{U} and v_k in \mathcal{V} . In addition, there is a node type mapping function $f_1 : \mathcal{U} \cup \mathcal{V} \rightarrow \mathcal{A}$ and an edge type mapping function $f_2 : \mathcal{E} \rightarrow \mathcal{R}$. Such a graph \mathcal{G} is called a bipartite graph.

Example 1 Given a search log, a query-item bipartite graph is built as shown in Figure 1, where $\mathcal{A} = \{Query, Item\}$ and $\mathcal{R} = \{Click\}$.

In a bipartite graph, we use the metapath and metapath instance to incorporate the neighboring node information into relevance matching. They are defined as follows:

Definition 2 Metapath and Metapath Instance in Bipartite Graph. Given a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, the metapath $P_i = a_1 \xrightarrow{r_1} a_2 \xrightarrow{r_2} \dots \xrightarrow{r_l} a_{l+1}$ ($a_j \neq a_{j+1}, 1 \leq j \leq l$) is a path from a_1 to a_{l+1} successively through r_1, r_2, \dots, r_l ($a_j \in \mathcal{A}, r_j \in \mathcal{R}$). The length of P_i is denoted as $|P_i|$ and $|P_i| = l$. For brevity, the set of all metapaths on \mathcal{G} can be represented in regular expression as $P^{\mathcal{G}} = (aa')^+(a|\varepsilon)|(a'a)^+(a'|\varepsilon)$ where $a, a' \in \mathcal{A}$ and $a \neq a'$. The metapath instance p is a definite node sequence instantiated from metapath P_i . All instances of P_i is denoted as $I(P_i)$, then $p \in I(P_i)$.

Example 2 As shown in Figure 1, an instance of metapath “Query-Item-Query” is “ $q_2-i_3-q_3$ ”.

Definition 3 k -order Relevance. Given a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, a function $F_{rel}^k : \mathcal{U} \times \mathcal{V} \rightarrow [0, 1]$ is called a k -order relevance function on \mathcal{G} if $F_{rel}^k(u_i, v_j) = G(\Phi(u_i), \Phi(v_j) | C_k)$, where $\Phi(\cdot)$ is a function to map each node to a representation vector, $G(\cdot)$ is the score function, $u_i \in \mathcal{U}, v_j \in \mathcal{V}$, and context information $C_k = \bigcup_{I_{P_i} \subseteq I(P_i), P_i \in P^{\mathcal{G}}, |P_i|=k} I_{P_i}$.

Many existing relevance matching models (Huang et al., 2013; Shen et al., 2014; Hu et al., 2014a) ignore context information C_k and only consider the sentences w.r.t. the query and item to be matched, which corresponds to 0-order relevance (for more details, please see the “Related Work” part in Appendix 4). We call it *context-free relevance matching* in this paper. Considering that both the 0-order neighbor (i.e., the node itself) and k -order neighbor ($k \geq 1$) are necessary for relevance matching, we argue that a reasonable mechanism should ensure that they can cooperate with each other. Then the research objective of our work is defined as follows:

Definition 4 Contextual Relevance Matching. Given a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$, the task of contextual relevance matching is to determine the context information C_k on \mathcal{G} and learn the score function $G(\cdot)$.

2.2 Overview

We propose a complete knowledge distillation framework (Figure 2), whose student model incorporates the context information, for contextual relevance matching in e-commerce product search. The main components of this framework are described as follows:

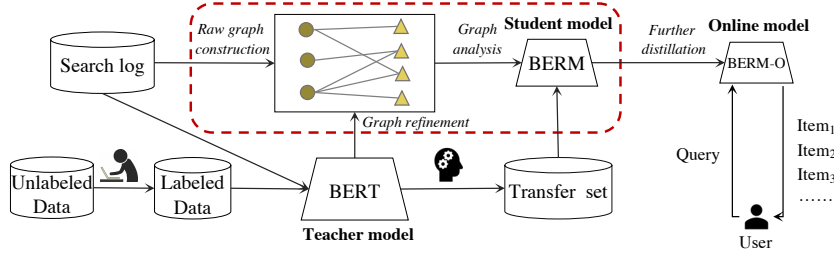


Figure 2: The e-commerce knowledge distillation framework proposed in our work. Three models are used in this framework: teacher model BERT, student model BERM, and online model BERM-O.

- **Graph construction.** We first construct a raw bipartite graph \mathcal{G} based on the search data collected from JD.com. Then we construct a knowledge-enhanced bipartite graph \mathcal{G}' with the help of BERT, which is fine-tuned by the human-labeled relevance data.
- **Student model design.** We design a novel student model BERM corresponding to the score function $G(\cdot)$ in Definition 4. Specifically, macro and micro matching embeddings are derived in BERM to capture the sentence-level and word-level relevance matching signal, respectively. Also, based on the metapaths “ $Q-I-Q$ ” and “ $I-Q-I$ ”, we design a node-level encoder and a metapath-instance-level aggregator to derive metapath embeddings.
- **Online application.** To serve online search, we conduct further distillation to BERM and obtain BERM-O, which is easy to be deployed online.

2.3 Bipartite Graph Construction

We introduce the external knowledge from BERT to refine the raw user behavior graph \mathcal{G} into a knowledge-enhanced bipartite graph \mathcal{G}' . The whole graph construction includes the following phases.

Fine-tuning BERT. We use the BERT model as the teacher model in our framework. BERT is pre-trained on a large text corpus and fine-tuned on our in-house data where the positive examples and negative examples are human-labeled and cover various item categories. The fine-tuned BERT is equipped with good relevance discrimination and thus acts as an expert in filtering noisy data. For each example pair p_i in the transfer set S_{transfer} , we use BERT to predict its score y_i as the training label of the student model BERM.

Behavior graph construction. The user behavior graph \mathcal{G} is built on the user search log over six months which records click behaviors and purchase

behaviors as well as their frequencies. Each edge in \mathcal{G} represents an existing click behavior or purchase behavior between the given query and item.

Knowledge-enhanced graph refinement. The click behavior edges are dense and highly noisy, so we leverage the fine-tuned BERT model to refine \mathcal{G} . Specifically, we retain all the raw purchase behavior edges, and meanwhile use the knowledge generated by the fine-tuned BERT to refine the click behavior edges. We set two thresholds α and β to determine which raw edges are removed and which new edges are added. This strategy helps remove the noise in user behaviors, and at the same time retrieve the missing but relevant neighbors which cannot be captured by user behaviors. To preserve important neighbors, for each anchor node, we rank its 1-hop neighbors with the priority of “purchase>high click>low click” and select the top two of them as the final neighbor list, i.e., the neighbor list of a query node Q is represented as $[I_{\text{top1}}, I_{\text{top2}}]$ and the neighbor list of a query node I is represented as $[Q_{\text{top1}}, Q_{\text{top2}}]$. The algorithm of graph construction is provided in Appendix A.

2.4 BERM Model

In this part, we describe BERM in detail, including 0-order relevance modeling, k -order relevance modeling, and overall learning objective.

2.4.1 0-order Relevance Modeling

The whole structure of BERM includes both the 0-order relevance modeling and k -order relevance modeling. This subsection introduces the 0-order relevance modeling which captures sentence-level and word-level matching signals by incorporating the macro matching embedding and micro matching embedding, respectively.

Macro and micro matching embeddings. Each word is represented by a d -dimensional embedding vector, which is trained by Word2Vec (Mikolov et al., 2013). The i -th word’s embedding of query

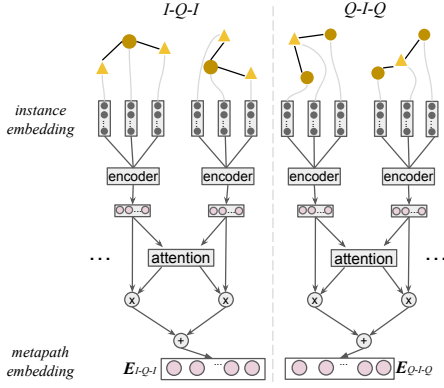


Figure 3: Calculation process of metapath embeddings.

Q (or item title I) is denoted as $E_Q^i \in \mathbb{R}^d$ (or $E_I^i \in \mathbb{R}^d$). To capture sentence-level and word-level matching signals, we employ macro matching embedding and micro matching embedding, respectively. For the macro matching embedding, taking query Q with l_Q words and item I with l_I words as examples, their macro embeddings $E_{\text{seq}}^Q, E_{\text{seq}}^I \in \mathbb{R}^d$ are calculated by the column-wise mean value of $E_Q \in \mathbb{R}^{l_Q \times d}, E_I \in \mathbb{R}^{l_I \times d}$:

$$E_{\text{seq}}^Q = \frac{1}{l_Q} \sum_{i=1}^{l_Q} E_Q^i, \quad E_{\text{seq}}^I = \frac{1}{l_I} \sum_{i=1}^{l_I} E_I^i. \quad (1)$$

For the micro matching embedding, we first build an interaction matrix $M_{\text{int}} \in \mathbb{R}^{l_Q \times l_I}$ whose (i, j) -th entry is the dot product of E_Q^i and E_I^j :

$$M_{\text{int}} = \{m_{\text{int}}^{i,j}\}_{l_Q \times l_I}, \quad m_{\text{int}}^{i,j} = \langle E_Q^i, E_I^j \rangle. \quad (2)$$

Then the micro matching embedding $E_{\text{int}} \in \mathbb{R}^{l_Q l_I}$ is the vectorization of M_{int} , i.e., $E_{\text{int}} = \text{vec}(M_{\text{int}})$.

2.4.2 k -order Relevance Modeling

The k -order relevance model contains a node-level encoder and a metapath-instance-level aggregator.

Node-level encoder. The input of the node-level encoder is node embeddings and its output is an instance embedding (i.e., the embedding of a metapath instance). Specifically, to obtain the instance embedding, we integrate the embeddings of neighboring nodes into the anchor node embedding with a mean encoder. Taking “ $Q-I_{\text{top1}}-Q_{\text{top1}}$ ” as an example, we calculate its embedding $E_{Q-I_{\text{top1}}-Q_{\text{top1}}} \in \mathbb{R}^d$ as follows:

$$E_{Q-I_{\text{top1}}-Q_{\text{top1}}} = \text{MEAN}(E_{\text{seq}}^Q, E_{\text{seq}}^{I_{\text{top1}}}, E_{\text{seq}}^{Q_{\text{top1}}}). \quad (3)$$

The metapath instance bridges the communication gap between different types of nodes and can be used to update the anchor node embedding from structure information.

Metapath-instance-level aggregator. The inputs of the metapath-instance-level aggregator are instance embeddings and its output is a metapath embedding. Different metapath instances convey different information, so they have various effects on the final metapath embedding. However, the mapping relationship between the instance embedding and metapath embedding is unknown. To learn their relationship automatically, we introduce the “graph attention” mechanism to generate metapath embeddings (Wu et al., 2021a; Liu et al., 2022). Taking metapath “ $Q-I-Q$ ” as an example, we use graph attention to represent the mapping relationship between “ $Q-I-Q$ ” and its instances. The final metapath embedding $E_{Q-I-Q} \in \mathbb{R}^d$ is calculated ($E_{I-Q-I} \in \mathbb{R}^d$ is calculated similarly) by accumulating all instance embeddings with attention scores $\text{Att}_1, \text{Att}_2, \text{Att}_3, \text{Att}_4 \in \mathbb{R}^+$:

$$E_{Q-I-Q} = \sigma(\text{Att}_1 \cdot E_{Q-I_{\text{top1}}-Q_{\text{top1}}} + \text{Att}_2 \cdot E_{Q-I_{\text{top1}}-Q_{\text{top2}}} + \text{Att}_3 \cdot E_{Q-I_{\text{top2}}-Q_{\text{top1}}} + \text{Att}_4 \cdot E_{Q-I_{\text{top2}}-Q_{\text{top2}}}), \quad (4)$$

where $\sigma(\cdot)$ is the activation function of LeakyReLU. Though Att_i can be set as a fixed value, we adopt a more flexible way, i.e., using the neural network to learn Att_i automatically. Specifically, we feed the concatenation of the anchor node embedding and metapath instance embedding into a one-layer neural network (its weight is $\mathbf{W}_{\text{att}} \in \mathbb{R}^{6d \times 4}$ and its bias is $\mathbf{b}_{\text{att}} \in \mathbb{R}^{1 \times 4}$) with a softmax layer, which outputs an attention distribution:

$$(\text{Att}_i)_{1 \leq i \leq 4} = \text{softmax}(E_{\text{concat}} * \mathbf{W}_{\text{att}} + \mathbf{b}_{\text{att}}), \quad (5)$$

$$E_{\text{concat}} = [E_{\text{seq}}^Q | E_{\text{seq}}^I | E_{Q-I_{\text{top1}}-Q_{\text{top1}}} | E_{Q-I_{\text{top1}}-Q_{\text{top2}}} | E_{Q-I_{\text{top2}}-Q_{\text{top1}}} | E_{Q-I_{\text{top2}}-Q_{\text{top2}}}], \quad (6)$$

The above process is shown in Figure 3.

Embedding fusion. By the 0-order and k -order relevance modeling, three types of embeddings are generated, including macro matching embedding ($E_{\text{seq}}^Q, E_{\text{seq}}^I \in \mathbb{R}^d$), micro matching embedding ($E_{\text{int}} \in \mathbb{R}^{l_Q l_I}$), and metapath embedding ($E_{Q-I-Q}, E_{I-Q-I} \in \mathbb{R}^d$). We concatenate them together and feed the result to a three-layer neural network (its weights are $\mathbf{W}_0 \in \mathbb{R}^{(4d+l_Q l_I) \times d}, \mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}, \mathbf{W}_3 \in \mathbb{R}^{d \times 1}$ and biases are $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^{1 \times d}, \mathbf{b}_3 \in \mathbb{R}^{1 \times 1}$), which outputs the final relevance estimation score \hat{y}_i :

$$\hat{y}_i = \text{Sigmoid}(E_3 * \mathbf{W}_3 + \mathbf{b}_3), \quad (7)$$

$$E_{j+1} = \text{ReLU}(E_j * \mathbf{W}_j + \mathbf{b}_j), \quad E_0 = E_{\text{all}}, \quad j = 0, 1, 2, \quad (8)$$

$$E_{\text{all}} = [E_{\text{seq}}^Q | E_{\text{seq}}^I | E_{\text{int}} | E_{Q-I-Q} | E_{I-Q-I}]. \quad (9)$$

2.4.3 Overall Learning Objective

We evaluate the cross-entropy error on the estimation score \hat{y}_i and label y_i (note that $y_i \in [0, 1]$ is the score of the teacher model BERT), and then minimize the following loss function:

$$\mathcal{L} = - \sum_{i=1}^{\tilde{n}} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i), \quad (10)$$

where \tilde{n} is the number of examples. We also analyze the complexities of BERT, BERM, and BERM-O in Appendix C.

3 Experiments

In this section, we present the offline and online experimental results of BERM*.

3.1 Experimental Setting

Datasets. We collect three datasets from the search platform of JD.com, including the “Electronics” category (Data-E), all-category (Data-A), and sampled all-category (Data-S). In the platform, there are mainly three different levels of item categories: Cid_1 (highest level, e.g., “Electronics”), Cid_2 (e.g., “Mobile phone”), and Cid_3 (lowest level, e.g., “5G phone”). Data-A, Data-S, and Data-E have different data distributions. Specifically, Data-A covers all first-level categories Cid_1 in the platform; Data-S is generated by uniformly sampling 5,000 items from Cid_1 ; Data-E only focuses on the category of “Electronics” in Cid_1 . Details of Data-E, Data-A, and Data-S are reported in Table 1.

For the training data S_{train} (also called S_{transfer}), the collected user behaviors include click and purchase. For the testing data S_{test} , whose queries are disjointed with those of S_{train} , we use human labeling to distinguish between relevant and irrelevant items. Specifically, editors are asked to assess the relevance scores between queries and items. In JD.com platform, the candidate set of relevance scores is $\{1, 2, 3, 4, 5\}$, where 5 means most relevant and 1 means least relevant. To simplify it, we use binary labeling including the positive label (i.e., 4 or 5) and negative label (i.e., 1, 2, or 3).

Evaluation Metrics. To measure the performance of baseline methods and our BERM, we use three kinds of evaluation metrics, including Area Under the receiver operating characteristic Curve (AUC), F1-score, and False Negative Rate (FNR).

*We provide the description of baselines, implementation details, and additional experiments in Appendix D.1, D.2, E (Code URL: <https://github.com/Young0222/EMNLP-BERM>).

Table 1: Statistics of the used datasets.

Set	Name	Data-E	Data-A	Data-S
S_{train}	# Example	6,369,396	174,863,375	11,397,439
	# Node _{query}	398,824	5,952,020	3,284,480
	# Node _{item}	728,405	49,517,217	1,307,557
	# Edge	5,070,460	159,205,320	7,525,355
	# Click	1,471,079,596	5,109,731,591	1,431,899,847
	# Purchase	33,285,887	322,151,488	118,495,170
S_{test}	# Example	30,563	39,743	39,743
	# Node _{query}	3,374	3,108	3,108
	# Node _{item}	16,137	30,097	30,097
	# Edge	18,988	30,661	30,661

Table 2: Comparisons on Data-E and Data-S. In each column, the best result is bolded and the runner-up is underlined. The symbol of “ \downarrow ” represents that the lower value corresponds to better performance. “I, II, III” represent the representation-focused, interaction-focused, and both-focused relevance matching models, respectively. “IV” represents the graph neural network models.

	Model	Data-E			Data-S		
		AUC	F1-score	FNR(\downarrow)	AUC	F1-score	FNR(\downarrow)
I	DSSM	0.6246	0.6923	0.9953	0.8219	0.8691	1.0000
	MVLSTM	<u>0.8602</u>	<u>0.8055</u>	0.3416	0.7877	0.8857	0.7802
	ARC-I	0.8343	0.7949	0.3857	0.6919	0.8750	0.9388
II	DRMM	0.6720	0.6891	0.7692	0.6781	0.8722	0.9401
	MatchPyramid	0.7826	0.7481	0.5615	0.7859	0.8786	0.8475
	ARC-II	0.8128	0.7864	0.4377	0.7606	0.8784	0.9076
	K-NRM	0.7462	0.7291	0.6510	0.7314	0.8733	0.9081
	DRMM-TKS	0.7678	0.7383	0.5462	0.7793	0.8789	0.7893
	Conv-KNRM	0.8369	0.7879	0.3469	0.8029	0.8789	0.8913
	ESIM	0.8056	0.7769	<u>0.3373</u>	0.7987	0.8623	1.0000
III	Duet	0.7693	0.7219	0.8173	0.7968	0.8754	0.9458
	BERT2DNN	0.8595	0.8037	0.3464	<u>0.8313</u>	<u>0.9061</u>	<u>0.4450</u>
IV	GAT	0.7526	0.7361	0.7529	0.7411	0.8746	0.9234
	GraphSAGE-Mean	0.7493	0.7330	0.7422	0.7406	0.8719	0.9119
	GraphSAGE-LSTM	0.7588	0.7509	0.6536	0.7529	0.8743	0.8652
	TextGNN	0.8310	0.8029	0.4525	0.8277	0.8779	0.7549
	GEPS	0.8405	0.8037	0.4892	0.8254	0.8794	0.6340
	BERM (ours)	0.8785	0.8256	0.2966	0.8758	0.9079	0.3625

The low value of FNR indicates the low probability of fetching irrelevant items, which is closely related to the user’s search experience. Therefore, we include it in the evaluation metrics.

3.2 Offline Performance

We compare BERM with 12 state-of-the-art relevance matching methods and 5 graph neural network models on our in-house product search data. The results are shown in Table 2. Because some baseline methods (e.g., DRMM and ESIM) have high time complexities, we use Data-E and Data-S for training and testing models.

As shown in Table 2, BERM outperforms all the baselines according to the metrics of AUC, F1-score, and FNR. More specifically, we have the following findings: 1) Compared to the second-

Table 3: Cases of e-commerce product search. “ y_i ” is the prediction score of the teacher model BERT and “ \hat{y}_i ” is the relevance estimation score of the student model BERM.

Query	Item title	Human labeling	y_i	\hat{y}_i
whistle	Li Ning whistle for basketball or volleyball game	Positive	0.9961	0.9681
women’s dance shoe	Sansha modern dance shoe P22LS (black, women)	Positive	0.9973	0.9908
violin adult	FineLegend 1/8 violin FLV1114	Positive	0.9950	0.9769
skating knee panels	RMT sports knee panels (black, L size)	Positive	0.9652	0.9841
DJI g1200a	DJI Mavic Mini unmanned aerial vehicle	Negative	0.0049	0.0514
Berkshire Hathaway Letters to Shareholders	The Snowball: Warren Buffett and the Business of Life	Negative	0.0258	0.0874
My brother called Shun Liu, ZHU SU JIN	Brothers: A Novel; Author: Yu Hua	Negative	0.1624	0.0263
nissan thermos cup	Disney thermos cup 500ML	Negative	0.4938	0.2513
java web exercises	JSP project development Case Full Record	Negative	0.5106	0.3111

best method MVLSTM (BERT2DNN), BERM surpasses it 1.83% (4.45%) according to AUC on Data-E (Data-S). Furthermore, BERM achieves the lowest value of FNR on both Data-E and Data-S. This implies that BERM can easily identify irrelevant items so that it can return a list of satisfactory items in the real-world scene. 2) The collected training data have imbalanced classes (i.e., the positive examples are far more than the negative examples), which poses a challenge to model learning. Most baselines are sensitive to class imbalance. Since BERM learns explicit node semantics by integrating the neighboring node information, our method is robust when the data are imbalanced.

3.3 Case Study

Apart from the above quantitative analysis, we conduct qualitative analysis based on some cases of e-commerce product search. For these cases, we list the query phrase, item title, human labeling, score of BERT, and score of BERM in Table 3. We have the following empirical conclusions: 1) Most of the student’s scores are close to the teacher’s, which indicates the success of the proposed knowledge distillation framework. 2) Some cases imply that context information is necessary for relevance matching. For example, for the query “nissan thermos cup”, the teacher model cannot explicitly judge whether or not the item entitled “Disney thermos cup 500ML” is relevant to it. With the help of context information in the query-item bipartite graph, BERM can recognize that this query is related to “nissan”, rather than “Disney”.

3.4 Deployment & Online A/B Testing

We conduct further distillation to BERM and obtain a lighter model BERM-O whose basic structure is a two-layer neural network. The process of further distillation is almost the same as the first knowledge distillation. The transfer set gener-

Table 4: Online performance of BERM-O under price sort mode and default sort mode.

Metric	Price sort mode		Default sort mode	
	Improvement	P-value	Improvement	P-value
UV-value	5.713%	3.20e-2	0.5013%	1.10e-1
UCVR	1.540%	7.81e-2	0.3058%	1.75e-2
CVR	1.829%	1.01e-2	0.1218%	1.60e-1
RPM	5.587%	3.03e-2	0.6886%	2.32e-2

ated by further distillation has graph-context labels. To further evaluate BERM-O’s performance in the real search scene, we deploy it to JD.com online search platform. On this platform, there are about one hundred million daily active users (DAU) and two billion items. It processes over 150 million search queries per day. The online baseline group BERT2DNN (Jiang et al., 2020) and control group BERM-O are deployed in a cluster, where each node is with 64 core Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz, 256GB RAM as well as 4 NVIDIA TESLA P40 GPU cards. For both groups, the only needed input data are queries and item titles, which can be easily caught from the online environment. Since BERM-O is lighter than BERT or BERM, deploying it to the online search chain requires less engineering work in the system.

Online results. We compare BERM-O with BERT2DNN (Jiang et al., 2020) which is our online baseline model using knowledge distillation without context information. The results of A/B testing are reported in Table 4. These results are from one observation lasting more than ten days. Four widely-used online business metrics are adopted 1) conversion rate (CVR): the average order number of each click behavior, 2) user conversion rate (UCVR): the average order number of each user, 3) unique visitor value (UV-value): the average gross merchandise volume of each user, and 4) revenue per mile (RPM): the average gross merchandise volume of each retrieval behavior. The results show

that BERM-O outperforms BERT2DNN in the platform according to all of the business metrics. For example, BERM-O significantly improves 5.7% (relative value) of UV-value under price sort mode.

4 Related Work

4.1 Classical Relevance Matching Models

The classical relevance matching models use the deep learning technique to learn vector representations containing the semantics of words or sequences. The prevailing methods are either representation-focused (e.g., DSSM (Huang et al., 2013), CDSSM (Shen et al., 2014), and ARC-I (Hu et al., 2014b)) or interaction-focused (e.g., MatchPyramid (Pang et al., 2016a), ARC-II (Hu et al., 2014b), and ESIM (Chen et al., 2017)). The representation-focused methods learn the low-dimensional representations of both sentences and then predict their relationship by calculating the similarity between the two representations. The interaction-focused methods learn an interaction representation of both sentences based on the calculation from word-level to sentence-level.

However, the above methods ignore the inherent context information contained in search logs (Qin et al., 2022; Roßbrucker, 2022). In this work, we incorporate the advantages of representation-based and interaction-based embeddings into BERM, which is focused on contextual relevance matching.

4.2 Transformer-style Models

More recently, Transformer-based models (Chen et al., 2021; Chi et al., 2021; Lin et al., 2021; Reid et al., 2021) have achieved breakthroughs on many NLP tasks and reached human-level accuracy. The representative models include BERT (Devlin et al., 2019), ERNIE (Sun et al., 2019b), and RoBERTa (Liu et al., 2019b). Additionally, GRMM (Zhang et al., 2021) and GHRM (Yu et al., 2021) use graph information to enforce the relevance matching model for information retrieval.

However, the multi-layer stacked Transformer structure in these models leads to high time complexity, so they are hard to deploy online. In this work, we use BERT to generate the supervised information of BERM and refine the noisy behavior data. Also, GRMM and GHRM are essentially different from ours in the definition of graphs. In their constructed graph, nodes are unique words and edges are the co-occurrent relationships. In this work, we leverage query phrases (or item ti-

ties) as nodes and user behaviors as edges, which is more suitable for the product search problem.

4.3 Online Knowledge Distillation Methods

Knowledge distillation is firstly proposed in (Hinton et al., 2015). Its main idea is to transfer the knowledge generated by a massive teacher model into a light student model. Because of the low complexity of the student model, it is easy to deploy the student model to the online platform. Considering the strong semantic understanding ability of BERT, some studies exploit the potential of BERT as the teacher model of knowledge distillation. Two types of design principles are general: isomorphic principle and isomeric principle. Specifically, the distillation methods that follow the isomorphic principle use the same model architecture for teacher and student models, such as TinyBERT (Jiao et al., 2020), BERT-PKD (Sun et al., 2019a), MTDNN (Liu et al., 2019a), and DistilBERT (Sanh et al., 2019). As a more advanced design principle, the isomeric principle uses different model architectures for teacher and student models, such as Distilled BiLSTM (Tang et al., 2019) and BERT2DNN (Jiang et al., 2020).

Although the above methods reduce the total time costs by learning a light student model, they ignore the context information in the real search scene. Our proposed knowledge distillation framework follows the isomeric principle and further integrates context information into the student model by bipartite graph embedding.

5 Conclusions and Future Work

In this paper, we propose the new problem of contextual relevance matching in e-commerce product search. Different from the previous work only using the 0-order relevance modeling, we propose a novel method of the k -order relevance modeling, i.e., employing bipartite graph embedding to exploit the potential context information in the query-item bipartite graph. Compared to the state-of-the-art relevance matching methods, the new method BERM performs robustly in the experiments. We further distill BERM into BERM-O and deploy BERM-O to JD.com online e-commerce product search platform. The results of A/B testing indicate that BERM-O improves the user’s search experience significantly. In the future, we plan to apply our method to other e-commerce applications such as recommender systems and advertisements.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (No. 61872207) and JD.com, Inc. Chaokun Wang is the corresponding author.

References

- Wei-Cheng Chang, Daniel Jiang, Hsiang-Fu Yu, Choon Hui Teo, Jiong Zhang, Kai Zhong, Kedarnath Kolluri, Qie Hu, Nikhil Shandilya, Vyacheslav Ievgrafov, et al. 2021. Extreme multi-label learning for semantic matching in product search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2643–2651.
- Pu-Chin Chen, Henry Tsai, Srinadh Bhojanapalli, Hyung Won Chung, Yin-Wen Chang, and Chun-Sung Ferng. 2021. A simple and effective positional encoding for transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2974–2988, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced lstm for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668.
- Zewen Chi, Li Dong, Shuming Ma, Shaohan Huang, Saksham Singhal, Xian-Ling Mao, Heyan Huang, Xia Song, and Furu Wei. 2021. mT6: Multilingual pretrained text-to-text transformer with translation pairs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1671–1683, Online and Punta Cana, Dominican 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.
- Chenhe Dong, Yaliang Li, Ying Shen, and Minghui Qiu. 2021. HRKD: Hierarchical relational knowledge distillation for cross-domain language model compression. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3126–3136, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 55–64.
- Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014a. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2042–2050.
- Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014b. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050.
- Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.
- Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338.
- Yunjiang Jiang, Yue Shang, Ziyang Liu, Hongwei Shen, Yun Xiao, Wei Xiong, Sulong Xu, Weipeng Yan, and Di Jin. 2020. Bert2dnn: Bert distillation with massive unlabeled data for online e-commerce search. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 212–221. IEEE.
- 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.
- Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Dynamic knowledge distillation for pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ye Lin, Yanyang Li, Tong Xiao, and Jingbo Zhu. 2021. Bag of tricks for optimizing transformer efficiency. In *EMNLP*.

- Hao Liu, Jindong Han, Yanjie Fu, Yanyan Li, Kai Chen, and Hui Xiong. 2022. Unified route representation learning for multi-modal transportation recommendation with spatiotemporal pre-training. *The VLDB Journal*, pages 1–18.
- Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv:1904.09482*.
- 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*.
- Yiqun Liu and Jiaxin Mao. 2020. " revisiting information retrieval tasks with user behavior models" by yiqun liu and jiaxin mao with martin vesely as coordinator. *ACM SIGWEB Newsletter*, pages 1–8.
- Jiaxin Mao, Zhumin Chu, Yiqun Liu, Min Zhang, and Shaoping Ma. 2019. Investigating the reliability of click models. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*, pages 125–128.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.
- Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1291–1299. ACM.
- Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3405–3415.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016a. Text matching as image recognition. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016b. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2793–2799.
- Chuan Qin, Kaichun Yao, Hengshu Zhu, Tong Xu, Dazhong Shen, Enhong Chen, and Hui Xiong. 2022. Towards automatic job description generation with capability-aware neural networks. *IEEE Transactions on Knowledge and Data Engineering*.
- Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. 2019. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. 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 5370–5381.
- Ahmad Rashid, Vasileios Lioutas, Abbas Ghaddar, and Mehdi Rezagholizadeh. 2021. Towards zero-shot knowledge distillation for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6551–6561, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Machel Reid, Edison Marrese-Taylor, and Yutaka Matsuo. 2021. Subformer: Exploring weight sharing for parameter efficiency in generative transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4081–4090, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Georg P Roßbrucker. 2022. State-of-the-art survey on web search. In *The Autonomous Web*, pages 1–24. Springer.
- 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*.
- Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd international conference on world wide web*, pages 373–374.
- Shuo Sun and Kevin Duh. 2020. CLIRMatrix: A massively large collection of bilingual and multilingual datasets for cross-lingual information retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4160–4170, Online. Association for Computational Linguistics.
- Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. 2019a. Patient knowledge distillation for bert model compression. 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 4323–4332.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

- Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.
- Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2835–2841.
- Junmei Wang, Min Pan, Tingting He, Xiang Huang, Xueyan Wang, and Xinhui Tu. 2020. A pseudo-relevance feedback framework combining relevance matching and semantic matching for information retrieval. *Information Processing & Management*, 57(6):102342.
- Xinle Wu, Dalin Zhang, Chenjuan Guo, Chaoyang He, Bin Yang, and Christian S Jensen. 2021a. Autocts: Automated correlated time series forecasting. *Proceedings of the VLDB Endowment*, 15(4):971–983.
- Yimeng Wu, Mehdi Rezagholizadeh, Abbas Ghaddar, Md Akmal Haidar, and Ali Ghodsi. 2021b. Universal-KD: Attention-based output-grounded intermediate layer knowledge distillation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7649–7661, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Chenyang Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 55–64. ACM.
- Song Xu, Haoran Li, Peng Yuan, Yujia Wang, Youzheng Wu, Xiaodong He, Ying Liu, and Bowen Zhou. 2021. K-PLUG: Knowledge-injected pre-trained language model for natural language understanding and generation in E-commerce. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1–17, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Xueli Yu, Weizhi Xu, Zeyu Cui, Shu Wu, and Liang Wang. 2021. Graph-based hierarchical relevance matching signals for ad-hoc retrieval. *CoRR*, abs/2102.11127.
- Chengxiang Zhai and John Lafferty. 2017. A study of smoothing methods for language models applied to ad hoc information retrieval. In *ACM SIGIR Forum*, volume 51, pages 268–276. ACM New York, NY, USA.
- Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural ir meets graph embedding: A ranking model for product search. *The World Wide Web Conference*.
- Yuan Zhang, Xiaoran Xu, Hanning Zhou, and Yan Zhang. 2020. Distilling structured knowledge into embeddings for explainable and accurate recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 735–743.
- Yufeng Zhang, Jinghao Zhang, Zeyu Cui, Shu Wu, and Liang Wang. 2021. A graph-based relevance matching model for ad-hoc retrieval. *CoRR*, abs/2101.11873.
- Jason Zhu, Yanling Cui, Yuming Liu, Hao Sun, Xue Li, Markus Pelger, Tianqi Yang, Liangjie Zhang, Ruofei Zhang, and Huasha Zhao. 2021. Textgnn: Improving text encoder via graph neural network in sponsored search. In *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 2848–2857. ACM / IW3C2.
- Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2020. Multimodal joint attribute prediction and value extraction for E-commerce product. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2129–2139, Online. Association for Computational Linguistics.

A Graph Construction Algorithm

We provide the complete algorithm of graph construction in Algorithm 1.

B Word Embedding in E-commerce Scene

In this part, we introduce the details of word embedding generation used in this work. In e-commerce scene, the basic representation of a query or an item is an intractable problem. On one hand, it is infeasible to represent queries and items as individual embeddings due to the unbounded entity space. On the other hand, product type names (like “iphone11”) or attribute names (like “256GB”) have special background information and could contain complex lexicons such as different languages and numerals. To address these problems, we adopt word embedding in BERM, which dramatically reduces the representation space. Also, we treat contiguous numerals, contiguous English letters, or single Chinese characters as one word and only retain the high-frequency words (such as the words occurring more than fifty times in a six-month search log) in the vocabulary. The final vocabulary is only in the tens of thousands, which saves memory consumption and lookup time of indexes by a large margin.

C Complexity Analysis

In this section, we analyze the time and space complexities of BERT (teacher model), BERM (student model), and BERM-O (online model).

C.1 Time Complexity

For the lookup operation on the static vocabulary table (i.e., a word embedding table whose size is n_w), the time complexities of BERT, BERM, and BERM-O are the same, i.e., $O(n_w)$. For the model calculation part, BERT uses Transformer networks. We denote the word embedding size, head number, network number, query length, and item length as d_1, h_1, k_1, l_Q, l_I , respectively. For the one-layer multi-head attention mechanism, the complexity of linear mapping (input part) is $O(\frac{1}{h_1}(l_Q + l_I)d_1^2)$, the complexity of attention operation is $O(h_1 l_Q^2 d + h l_I^2 d_1)$, and the complexity of linear mapping (output part) is $O((l_Q + l_I)d_1^2)$. Therefore, the total model calculation complexity of k_1 -layer BERT is $O(\frac{h_1+1}{h_1}(l_Q + l_I)d_1^2 k_1 + (l_Q^2 + l_I^2)h_1 d_1 k_1)$. For the student model BERM, we denote the word em-

Algorithm 1 Bipartite Graph Construction

Input: Thresholds α and β ; Collected dataset $S_{\text{input}} = \{p_i\}_{\tilde{n}}$ (note that \tilde{n} is the number of examples); Raw user behavior graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R})$ where $\mathcal{E} = \{e_1, \dots, e_m\}$, $\mathcal{A} = \{Query, Item\}$, $\mathcal{R} = \{Click, Purchase\}$.

Output: Transfer set $S_{\text{transfer}} = \{p_i; y_i\}_{\tilde{n}}$ where y_i is the training label of query-item pair p_i ($y_i \in [0, 1]$); Refined bipartite graph $\mathcal{G}' = (\mathcal{U}, \mathcal{V}, \mathcal{E}', \mathcal{A}, \mathcal{R})$.

- 1: Initialize \mathcal{E}' : $\mathcal{E}' = \mathcal{E}$.
 - 2: Fine-tune BERT on the human-labeled data.
 - 3: Use the fine-tuned BERT to predict on S_{input} and then obtain $S_{\text{transfer}} = \{p_i; y_i\}_{\tilde{n}}$.
 - 4: **for** p_i, y_i in S_{transfer} **do**
 - 5: Build an edge between the pair p_i and denote it as $e_i = \text{edge}(p_i)$.
 - 6: **if** $e_i \in \mathcal{E}$ **then**
 - 7: **if** $f_2(e_i) = Purchase$ **then**
 - 8: continue;
 - 9: **else if** $y_i < \alpha$ **then**
 - 10: $\mathcal{E}' = \mathcal{E}' \setminus \{e_i\}$
 - 11: **end if**
 - 12: **else if** $y_i > \beta$ **then**
 - 13: $\mathcal{E}' = \mathcal{E}' \cup \{e_i\}$
 - 14: **end if**
 - 15: **end for**
-

bedding size, hidden size, and network number as d, h_2, k_2 , respectively. The complexity of calculating micro matching embedding is $O(l_Q l_I d)$, which is far more than that of calculating micro matching embedding. The complexity of k_2 -layer DNN is $O(h_2 d k_2)$. Therefore, the total model calculation complexity of k_2 -layer BERM is $O(l_Q l_I d + h_2 d k_2)$. For the online model BERM-O, we denote the word embedding size, hidden size, and network number as d_3, h_3, k_3 , respectively. The model calculation complexity of k_3 -layer BERM-O is $O(d_3 h_3 k_3)$. Note that the complexity of BERM-O is independent of l_Q and l_I because BERM-O only receives sentence embeddings and does not calculate word-level matching signals. Based on the above analysis, we can conclude that BERM is more efficient than BERT and meanwhile BERM-O has more advantages than BERM on time complexity.

C.2 Space Complexity

The storage of the static vocabulary table takes up the majority of the total space storage. Therefore, the space complexities of BERT, BERM, and BERM-O are the same, i.e., $O(n_w d)$.

D Details of Experimental Setups

D.1 Baselines

The model BERM is compared with some state-of-the-art models. Like BERM, these models are used as the student model of the proposed knowledge dis-

tillation framework. We adopt the hyper-parameter settings recommended by the original papers for all the methods. According to the formulation process of embedding, these methods can be divided into the following three types:

- Three representation-focused relevance matching methods: DSSM (Huang et al., 2013), MVLSTM (Wan et al., 2016), and ARC-I (Hu et al., 2014b). They learn the low-dimension representations of both sentences w.r.t. a query and an item, and then predict their relationship by calculating the similarity (such as cosine similarity) of representations.
- Seven interaction-focused relevance matching methods: DRMM (Guo et al., 2016), Match-Pyramid (Pang et al., 2016b), ARC-II (Hu et al., 2014a), K-NRM (Xiong et al., 2017), DRMM-TKS (Guo et al., 2016), Conv-KNRM (Xiong et al., 2017), and ESIM (Chen et al., 2017). They learn an interaction representation of both sentences based on the interaction calculation from word-level to sentence-level.
- Two integrated relevance matching methods: Duet (Mitra et al., 2017) and BERT2DNN (Jiang et al., 2020). They combine the features of the above two types of methods into themselves.
- Five graph neural network models: GAT (Veličković et al., 2018), GraphSAGE-Mean (Hamilton et al., 2017), GraphSAGE-LSTM (Hamilton et al., 2017), TextGNN (Zhu et al., 2021), and GEPS (Zhang et al., 2019). They aggregate the neighbor information from the query graph or item graph to update the embedding of the anchor node.

D.2 Implementation Details

Here we introduce the implementation details of the whole knowledge distillation framework as follows:

- **Teacher model.** For the teacher model, we adopt BERT-Base* with a 12-layer ($k_1=12$) Transformer encoder where the word embedding size d_1 is 768 and head number h_1 is 12. We pre-train BERT-Base on a human-labeled dataset with 380,000 query-item pairs. The fine-tuned BERT-Base is then used as an expert to refine the noisy click behavior data from S_{train} . The

*<https://github.com/google-research/bert>

refinement rule is: if the prediction score y_i of BERT-Base is less than α (the default value of α is 0.3), then the raw edge is deleted; if the score is larger than β (the default value of β is 0.7), then a new edge is added.

- **Student model.** For the student model, we adopt the proposed BERM model. We implement BERM in TensorFlow 2.0 with the high-level Estimator API. For each input query phrase Q or item title I , we split it into several words and then truncate or pad its length to 10 or 65 words (i.e., $l_Q=10, l_I=65$). Each word embedding is acquired by the lookup operation on a static vocabulary table whose total size n_w is 39,846. This table is generated by pre-training two billion search data with the tool of Word2Vec. The size d of pre-trained embeddings or trained embeddings is 128.
- **Training details.** We use Lazy-Adam as the optimizer and its learning rate is 0.001. To reduce the overfitting of the training data, we use L2 regularization on each layer of neural networks. For Data-E, we set the training epoch as 20. For Data-A and Data-S, we set the training epoch as 3.

E Additional Experiments

We conduct some additional experiments, including ablation studies (Section E.1) and sensitivity analysis (Section E.2). In these experiments, we adopt Data-E and Data-A consistently.

E.1 Ablation Study

E.1.1 Integration of Embeddings

There are three types of components in the complete BERM: the representation-based embeddings $E_{\text{seq}}^Q, E_{\text{seq}}^I$, interaction-based embedding E_{int} , and metapath embeddings E_{Q-I-Q}, E_{I-Q-I} . To further examine the importance of each component in the final embedding of BERM, we remove one or two components from it (Equation 9) at a time and examine how the change affects its overall performance.

The corresponding results on Data-E and Data-A are reported in Table 5 and 6. We have the following empirical observation and analysis:

- In general, the both-component setting outperforms the single-component setting but is worse than the triple-component setting (i.e., BERM).

Table 5: Ablation study on Data-E. In each column, the best result is bolded.

Model	Data-E		
	AUC	F1-score	FNR(\downarrow)
E_{seq}^Q, E_{seq}^I	0.8537	0.8044	0.3560
E_{int}	0.8595	0.8037	0.3464
E_{Q-I-Q}, E_{I-Q-I}	0.8430	0.8173	0.2995
$E_{seq}^Q, E_{seq}^I, E_{int}$	0.8638	0.8086	0.3331
$E_{int}, E_{Q-I-Q}, E_{I-Q-I}$	0.8761	0.8221	0.2758
$E_{seq}^Q, E_{seq}^I, E_{Q-I-Q}, E_{I-Q-I}$	0.8656	0.8190	0.2922
BERM	0.8785	0.8256	0.2966

Table 6: Ablation study on Data-A. In each column, the best result is bolded.

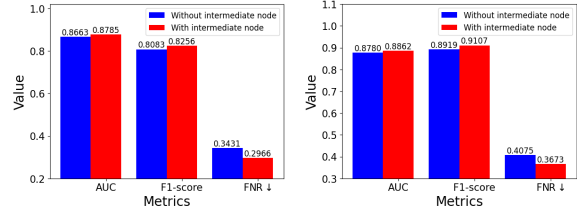
Model	Data-A		
	AUC	F1-score	FNR(\downarrow)
E_{seq}^Q, E_{seq}^I	0.8067	0.8660	0.3697
E_{int}	0.8289	0.9084	0.4459
E_{Q-I-Q}, E_{I-Q-I}	0.8500	0.9114	0.4110
$E_{seq}^Q, E_{seq}^I, E_{int}$	0.8776	0.9070	0.4743
$E_{int}, E_{Q-I-Q}, E_{I-Q-I}$	0.8824	0.9099	0.3705
$E_{seq}^Q, E_{seq}^I, E_{Q-I-Q}, E_{I-Q-I}$	0.8750	0.9094	0.3753
BERM	0.8862	0.9107	0.3673

It demonstrates that different components in BERM have different positive effects on the overall performance and they cannot replace each other.

- The introduction of k -order relevance modeling can bring stable advancement to each 0-order relevance model. For example, the combination of “ $E_{seq}^Q, E_{seq}^I, E_{Q-I-Q}, E_{I-Q-I}$ ” surpasses the combination of “ E_{seq}^Q, E_{seq}^I ” 6.83% according to the metric of AUC on Data-A. This demonstrates that applying metapath embedding to relevance matching can make effective use of the neighboring nodes’ information in the user behavior graph.

E.1.2 Effect of the Intermediate Node

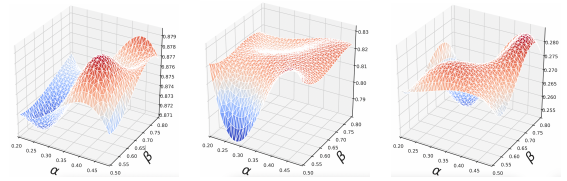
The metapath defined in BERM includes the intermediate node. To further investigate the effect of the intermediate node, we compare the performances of BERM with the intermediate node (i.e., “ $Q-I-Q$ ” and “ $I-Q-I$ ”) and BERM without the intermediate node (i.e., “ $Q-Q$ ” and “ $I-I$ ”) on Data-E and Data-A in Figure 4. We observe that BERM with the intermediate node performs better than the other one. We infer that the intermediate node has strong semantic closeness to the anchor node and thus it is helpful for accurate semantic recognition.



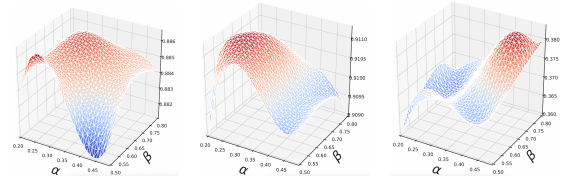
(a) Data-E

(b) Data-A

Figure 4: Effect of the intermediate node. The red (blue) bar represents BERM with (without) the intermediate node.



(a) AUC of Data-E (b) F1-score of Data-E (c) FNR of Data-E



(d) AUC of Data-A (e) F1-score of Data-A (f) FNR of Data-A

Figure 5: Effect of different values of α and β . (a), (b), and (c) are the results of Data-E; (d), (e), and (f) are the results of Data-A. The red (blue) color corresponds to the high (low) value.

E.2 Sensitivity Analysis

E.2.1 Thresholds α and β

In BERM, α decides how many edges of the noisy click behavior should be deleted and β decides how many hidden useful edges should be retrieved. To investigate the sensitivity of α and β , we conduct experiments with 16 different hyper-parameter settings where α ranges from 0.2 to 0.5 and β ranges from 0.5 to 0.8. We apply the three-order curve interpolation method to show the final results in Figure 5. In general, the results of BERM are robust to the change of hyper-parameter α and β on either Data-E or Data-A. For example, the maximum error of AUC is no more than 1%. So we conclude that user behaviors play a major role in the performance of BERM and the knowledge from BERT provides auxiliary effects for it.

Table 7: Effect of different neighbor selection strategies on Data-E.

Rate	Click+BERT’s score			Purchase+BERT’s score		
	AUC	F1-score	FNR(↓)	AUC	F1-score	FNR(↓)
$\lambda = 0.0$	0.8785	0.8256	0.2966	0.8785	0.8256	0.2966
$\lambda = 0.2$	0.8779	0.8237	0.2834	0.8777	0.8236	0.2810
$\lambda = 0.4$	0.8770	0.8200	0.3198	0.8756	0.8214	0.3068
$\lambda = 0.6$	0.8670	0.8085	0.4000	0.8696	0.8045	0.3694
$\lambda = 0.8$	0.8679	0.8101	0.3901	0.8660	0.8105	0.3262
$\lambda = 1.0$	0.8656	0.8091	0.3850	0.8671	0.8109	0.3727

Table 8: Effect of different neighbor selection strategies on Data-A.

Rate	Click+BERT’s score			Purchase+BERT’s score		
	AUC	F1-score	FNR(↓)	AUC	F1-score	FNR(↓)
$\lambda = 0.0$	0.8862	0.9107	0.3673	0.8862	0.9107	0.3673
$\lambda = 0.2$	0.8849	0.9113	0.3794	0.8830	0.9117	0.3831
$\lambda = 0.4$	0.8821	0.9112	0.4016	0.8818	0.9100	0.4131
$\lambda = 0.6$	0.8779	0.9068	0.4793	0.8783	0.9064	0.4844
$\lambda = 0.8$	0.8802	0.9076	0.4676	0.8790	0.9084	0.4683
$\lambda = 1.0$	0.8792	0.9077	0.4671	0.8787	0.9079	0.4716

E.2.2 Threshold k

Here we evaluate the effect of k on the performance of BERM by sampling neighboring nodes with different hops from the bipartite graph. The comparison results on Data-E and Data-S are shown in Figure 6. We can see that the BERM with $k=2$ achieves the best performance among them. When k is too large such as $k=5$, many distant neighbors are aggregated into the anchor nodes, then it leads to the performance degradation of BERM due to lots of noise gathering in these distant neighbors. Therefore, we conclude that 2-order relevance matching modeling is the optimal choice for our e-commerce scene.

E.2.3 Selection of Neighbor Structure

In BERM, the selection of neighbor structure directly affects which context information is transmitted to the anchor node. A good selection strategy can aggregate valuable neighboring node information to enrich the anchor node’s representation. To investigate the effect of different neighbor structure selection strategies on BERM and seek a relatively optimal solution, we use different values of hyper-parameter λ to control the ratio between user behavior and BERT’s score. Specifically, we calculate a new score $\text{Score}_{\text{new}}(Q, I) = \lambda \cdot \text{User}(Q, I) + (1-\lambda) \cdot \text{Score}(Q, I)$ where $\text{User}(Q, I)$ is the user behavior feature (e.g., for click behavior, $\text{User}(Q, I)=1$ if click behavior happens between query Q and item I). The addition and dele-

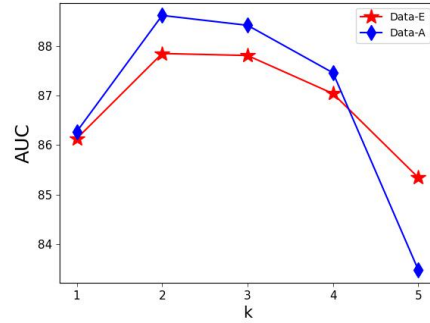


Figure 6: The effect of different k .

tion of edges refer to $\text{Score}_{\text{new}}(Q, I)$, rather than $\text{Score}(Q, I)$. We report the results with different λ in Table 7 and 8. From them, we can conclude that:

- Using BERT’s score is better than using user behaviors for the selection of neighbors. Therefore, the value of AUC or F1-score gradually decreases with the increase of λ ; the value of FNR increases with the increase of λ . The optimal λ is located in the interval $[0.0, 0.2]$.
- According to the metric of FNR, the purchase behavior is better than the click behavior on Data-E. The reason for it is that purchase behaviors reveal more accurate semantic relevance information than click behaviors. However, the purchase behavior is worse than the click behavior on Data-A. We guess that it is caused by the sparsity of purchase behaviors in the dataset of all categories.

Accelerating the Discovery of Semantic Associations from Medical Literature: Mining Relations Between Diseases and Symptoms

Alberto Purpura
IBM Research Europe
alp@ibm.com

Francesca Bonin
IBM Research Europe
fbonin@ie.ibm.com

Joao H. Bettencourt-Silva
IBM Research Europe
jbettencourt@ie.ibm.com

Abstract

Medical literature is a vast and constantly expanding source of information about diseases, their diagnoses and treatments. One of the ways to extract insights from this type of data is through mining association rules between such entities. However, existing solutions do not take into account the semantics of sentences from which entity co-occurrences are extracted. We propose a scalable solution for the automated discovery of semantic associations between different entities such as diseases and their symptoms. Our approach employs the UMLS semantic network and a binary relation classification model trained with distant supervision to validate and help ranking the most likely entity associations pairs extracted with frequency-based association rule mining algorithms. We evaluate the proposed system on the task of extracting disease-symptom associations from a collection of over 14M PubMed abstracts and validate our results against a publicly available known list of disease-symptom pairs.

1 Introduction

Scientific literature is a valuable resource for accelerating scientific discovery in several fields, from computer science to physics, and healthcare (Kumar and Tipney, 2014). However, the overwhelming amount of articles that need to be inspected requires extensive computational approaches as well as modeling knowledge in an appropriate machine readable form. Many efforts have been recently tackling the problem of transforming the unstructured knowledge from scientific papers to knowledge graphs that enable the extraction of actionable insights (Hou et al., 2019; Yadav et al., 2020; Park et al., 2021).

Due to the phenomenal growth of PubMed and MedLine publications (Bretonnel and Lawrence, 2008), the medical domain would particularly benefit from the creation of comprehensive knowledge

graphs. Extracting relations between entities is particularly valuable in the medical and biomedical domains where scientists need to extract semantic relations between medical concepts, such as protein and protein, gene and protein, drug and drug, and drug and disease. These relations can be extracted from biomedical literature available from various sources and have already been made accessible in different databases such as BioGRID (Stark et al., 2006) or PDID (Wang et al., 2016). The extraction of these associations from biomedical literature, however, is often time consuming and computationally expensive. Hence, these databases become quickly outdated if they are not updated at the same rate as new scientific discoveries are published. In fact, to the best of our knowledge, the usage of fully automated tools for the extraction of information from these sources is very limited.

In this paper, we propose an efficient end-to-end pipeline for the extraction of semantic relations between medical concepts. We evaluate our approach on disease-symptom associations discovery, but it can also be applied to other relation types and use cases. While disease-symptom information is widely published in medical bibliography, mining such information from literature, electronic health records, or even from user generated content, can accelerate the detection of new symptoms, diseases or variants. Early detection is particularly important in public health surveillance, both in detecting new pandemics (as was the case for COVID-19), identifying new symptoms associated with known diseases (also seen in COVID-19), or even for detecting the resurgence of disease outbreaks in certain countries – as was the case for the Ebola outbreak of 2014 in West Africa. Being able to recognize relations between medical concepts also means that biomedical or clinical texts can be automatically processed at scale, resulting in tools to support decision-making, clinical trial screening, and pharmacovigilance (Yadav et al., 2020).

The proposed pipeline operates similarly to a search engine and can therefore return up-to-date information as its database of documents can be updated in near real time. Our main contributions are:

- a scalable and easily updatable Elasticsearch-based solution to store and query annotated medical literature documents;
- an efficient association rules discovery system based on (i) an association rule mining algorithm, (ii) UMLS semantic network information and (iii) a binary relation classification model trained with distant supervision;
- a solution for human-in-the-loop verification and interpretation of the discovered disease-symptoms associations.

In addition, our solution uses open source libraries and models, and does not require expensive annotation efforts.

An Industry Perspective. With the advance of health informatics, several applications are being developed in the healthcare industry. Automated diagnosis applications as well as symptom checkers are widely used and especially important in low-resource countries to ensure remote medical assistance (Morita et al., 2017). Similarly, health insurance providers and public health organizations are increasingly interested in preventative care to detect early disease onset or complications before patients become expensive and risky to treat.

One of the issues for such application is the collection of information for specific populations. The system proposed in this paper, because of its scalability and ability to interrogate more than 14M abstracts, can represent a useful resource that allows near real time searches to be performed for relation discovering.

2 Related Work

Relation extraction from biomedical text, clinical discharge notes and medical articles have been widely investigated. Among the early systems, Chen et al. (2008) proposed a combination of text mining and association rules to determine the association of drugs and diseases. More recently, increasing numbers of machine learning-based approaches have been developed exploiting supervised learning and feature engineering. Many of them have looked at identifying modifiers related to important clinical entities, such as medication

features (Jon D. and Min, 2010; Pathak et al., 2015). An interesting relation extraction task was proposed in the 2010 i2b2/VA challenge (Uzuner et al., 2011). Task organizers released an annotated dataset with medical concepts, assertions and relations, and participants were asked to extract both concepts and assertion as well as specific relations between relevant clinical entities in text. All the top-ranked systems used machine learning-based methods with extensive feature engineering. For example, Grouin et al. (2010) proposed a Support Vector Machine (SVM) based system with additional rules to capture linguistic patterns of relations, while De Bruijn et al. (2011) investigated machine learning using large-dimensional features derived from both textual information and other external sources. Differently from those approaches, we focus on disease-symptom associations, which are useful for a wide variety of healthcare applications and our approach does not require costly manual annotations or extensive feature engineering. More recently, deep learning models have also been applied to solve the same task. Shah et al. (2019) introduce a concept association mining framework based on word embeddings learned through neural networks allowing diseases and related symptoms to be visualized in chronological order. Zhang and Lu (2019) have used a semi-supervised approach based on variational autoencoder for biomedical relation extraction where a multi-layer Convolutional Neural Network (CNN) was used together with bidirectional long short-term memory networks (Bi-LSTMs) to encode drug-drug, protein-protein and chemical-protein interactions. Similar tasks were carried out by Yadav et al. (2020) who introduced a multi task learning framework leveraging a structured self-attentive network together with adversarial learning, and their approach covers also the task of medical concept relation extraction. Our approach explores a different solution, combining pattern mining with NLP algorithms and leveraging more than 14M PubMed abstract, to exploit the vastness of publications to its fullest.

3 Proposed System

The proposed system for the discovery of semantic entity associations is divided into three parts: 1) a sentence annotation system, 2) an Elasticsearch¹ index and 3) a querying and filtering component. The latter can be further divided into three units: (i)

¹<https://www.elastic.co>.

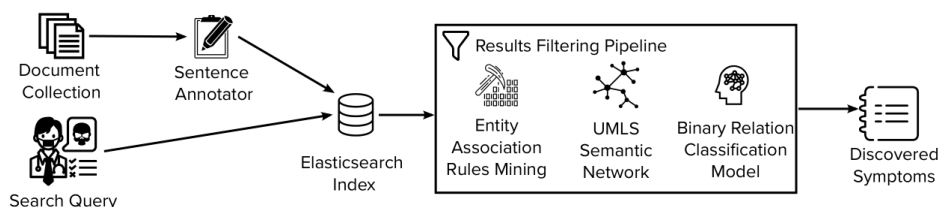


Figure 1: Proposed pipeline architecture.

an Association Rules Mining (ARM) algorithm, (ii) a Unified Medical Language System (UMLS) (Bodenreider, 2004) semantic network-based association rules filter and (iii) a Binary semantic Relation Classification (BRC) model. When deployed, a user can query our system providing a disease name or its UMLS unique identifier and receive a ranked list of symptoms that are likely associated to it, together with a few sentences from the indexed data that motivate the discovered disease-symptom associations. The system facilitates the addition of new documents – such as newly published papers – with the automated annotation of the new textual data and a call to the index API to insert the new records to the index without any updates to the results filtering pipeline. A diagram of the proposed system is depicted in Figure 1.

Documents Annotation and Indexing. When a new document is received, we first split it into sentences and annotate each of them with UMLS entity codes using Scispacy’s (Neumann et al., 2019) Natural Language Processing (NLP) pipeline. Scispacy is one of the most efficient and popular NLP annotation libraries for biomedical data and allows for a straightforward integration with the large ecosystem of Python libraries. Its performance is similar to other popular libraries such as MetaMap (Aronson, 2001) but it has a lower inference time (Neumann et al., 2019). For the above reasons, we decide to employ it as the annotation component for our pipeline. Scispacy’s entity linker also recognizes mentions of entities from the UMLS database in each of the sentences of the input text. Each of the entities recognized by the annotator has a corresponding UMLS semantic type, such as diseases, symptoms, or anatomy terms, among others. Here, we are going to focus only on the former two entity semantic types – i.e. diseases and symptoms – however, the proposed solution could discover relations between any entity types with minimal adaptations. Once a sentence has been annotated, we store it in an Elasticsearch index. There, we associate different tags to it indicating the UMLS unique identifiers of

the entities mentioned in it. We then employ these tags to efficiently retrieve relevant sentences mentioning a certain disease and its symptoms. We use the *keyword* data type from Elasticsearch to store a list of entity identifiers for each sentence record. This strategy allows us to efficiently compute the support (Agrawal and Srikant, 1994) of different diseases and disease-symptom pairs.² To summarize, when receiving a new textual document, this component of our pipeline: (1) splits into sentences, (2) adds metadata to each sentence related to the spans of each entity mentioned in it, i.e. a disease, drug or symptom name and their UMLS identifiers, (3) store the sentence and metadata into an Elasticsearch index. Finally, we are aware that the output of this component could contain errors, e.g. mislabeled entities. We protect from them by collecting data from a large number of documents and relying on approaches that are robust to outliers as described in the following sections.

Association Rules Mining (ARM). To discover associations between annotated entity pairs – e.g. a disease and its symptoms – we first query our index to retrieve all sentences mentioning the given disease. For each sentence where the given disease is mentioned, we also obtain a list of other entity identifiers that co-occur with the same disease in that sentence. We then extract all frequent itemsets of size two containing the given disease and a symptom among all possible combinations between any co-occurring pair. For each candidate symptom entity we then compute the support of the itemset of size two containing the given disease by querying the Elasticsearch index. Finally, we compute the confidence of the association rule: $conf(D \Rightarrow C) = supp(D, C) / supp(D)$, where D indicates a disease, C a candidate symptom entity, $supp(\cdot)$ the operation to compute the support of an itemset and $D \Rightarrow C$ an association rule between a disease D and another entity C . We em-

²The support of an itemset is defined as the number of times it occurs in a dataset. An itemset could contain a single entity (e.g. a disease) or more than one (e.g. a disease-symptom pair).

ploy this score to rank disease-entity pairs that are most likely to be associated based on the statistical properties of the collected data. This solution for ARM is similar in principle to the popular Pointwise Mutual Information (PMI) (Church and Hanks, 1990) measure of association between words, but allows the discovery of associations between groups of entities larger than two, which can be useful for certain applications.

We also experimented with computing the Lift of each association rule, defined as $Lift(C \Rightarrow D) = conf(D \Rightarrow C) / supp(C)$. However, we found that this metric – which is normalized with respect to the frequency of each symptom – led to lower performances than the former. This is likely due to the fact that it does not take into account the frequency of each candidate symptom alone which is some important information when dealing with noisy annotations coming from text.

Frequency-based measures, however, are not precise enough to recognize all *semantic* disease-symptoms associations in our data. For this reason, we apply two further steps based on the information contained in the UMLS Semantic Network and on our BRC model.

UMLS Semantic Network Based Pruning.

From the perspective of our ARM approach, any entity co-occurring with a disease is considered a potential relevant match. To remove association rules involving entities that do not represent a symptom or are not semantically related to a disease, we apply a filtering step based on the information contained in the UMLS semantic network. This is a network of semantic types such as *Disease*, *Symptom* or *Gene*, among others.

Each of these semantic types is associated to a set of UMLS entity identifiers and allows us to identify entities typically recognized as symptoms by the medical community and the relations between them and other entities in the UMLS ontology. In principle, any symptom associated to the *disease* semantic type can be associated to the disease that we are interested in analyzing. The semantic network, however, only yields semantic information between broad semantic concepts that could all be potentially related to each other and does not provide any specific information regarding a particular disease or symptom association. Therefore, this filtering stage allows us to distinguish between a gene or a body part and a symptom that could occur with *any* disease and to remove these *semantically incor-*

rect associations. For this reason, we employ it in association with the former statistical-based entity strategy and with the BRC model described below to filter out some of the candidate entity pairs.

Binary Relation Classification (BRC). The final step we propose to extract and rank the most likely symptoms for a disease is the BRC model. We randomly sample n sentences (set to 500 in our experiments) where a disease is mentioned together with each of the candidate symptoms and feed them to our BRC model. This model is trained to predict whether two entities in a sentence are semantically related or not. For example, in the sentence: “We conclude that the ability of stress testing to predict *coronary-artery disease* is limited in a heterogeneous population in which the prevalence of disease can be estimated through classification of *chest pain* and the sex of the patient.” the two entities *chest pain* (symptom) and *coronary-artery disease* (disease) are semantically related because the sentence is expressing a concept that puts the entities in relation to each other. We are not interested in a specific relation type between the two entities (a symptom-disease relation can be *manifestation-of*, *evaluation-of*, *diagnoses of* according to the UMLS semantic network). Instead, we are interested in recognizing sentences expressing *any* semantic relation between two given entities. The confidence score returned by this component of our pipeline is equal to the ratio of the examined sentences where the same entity pair is classified as semantically related out of the total number of examined sentences where both entities occur (n). For instance, we consider 500 random sentences from our index where a disease like “Asthma” is mentioned together with one of its candidate symptoms such as “Dyspnea”. We then classify the relation between these two entities in each of the 500 sentences. We finally compute the BRC association score as the number of sentences where the two entities were classified as related over the total of 500.

The architecture of the proposed model is depicted in Figure 2. We employ a BERT transformer model trained on PubMed abstracts – PubMedBERT (Gu et al., 2021) – available in the Hugging Face library and fine-tune it on the binary classification task on a training dataset we generated automatically with distant supervision. The input to the model is a sentence with entity mentions tags like $\langle e1 \rangle \dots \langle e1 \rangle$ and $\langle e2 \rangle \dots \langle e2 \rangle$ surround-

ing the surface forms of the pair of entities we are interested in. Next, we encode the input sentence with PubMedBERT and then take the representation of the $\langle e1 \rangle$ and $\langle e2 \rangle$ tokens from the last layer of the encoder and concatenate them. This representation of the entities in the input sentence is then fed to a feed-forward neural network with a softmax activation that outputs the probabilities of the two input entities being semantically related to each other given the input sentence. To generate the training data for our model we consider the sentences stored in the aforementioned Elasticsearch index. We select a subset of 300K sentences containing pairs of entities that are related to each other according to the UMLS semantic network – we used 90% of them for training and 10% for validation. These sentences are automatically labeled as containing a relation between a UMLS semantically related entity pair if the same entities are also syntactically related. All other instances from the aforementioned group containing syntactically unrelated entity pairs are considered as negative samples. We say that two entities are *syntactically* related if the root verb of the sentence appears in the shortest path connecting them in the sentence dependency tree. We observed empirically that combining this simple syntactic rule with the information from the UMLS semantic network yields results of a sufficiently high quality to train our BRC model. Other strategies similar to ours are also frequently employed for the creation of relation classification datasets (Smirnova and Cudré-Mauroux, 2018). We fine-tune the transformer model for 10 epochs using Cross-Entropy loss, batch size 64 and learning rate of $2e-5$. After training, our model achieved an F1 Score of 0.94 on our randomly sampled validation set.

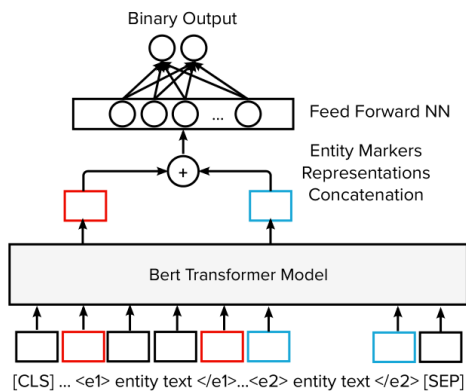


Figure 2: Binary Relation Classification (BRC) model.

Results ranking. The disease-symptoms pairs discovered filtered by the UMLS semantic network are finally ranked in decreasing order of relevance with respect to a combination of the output scores of the ARM and BRC models: $\text{Score}(d, s) = \alpha \cdot \text{BRC}(d, s) + \beta \cdot \text{ARM}(d, s)$ where α and β are parameters optimized using a held out validation set of known entity pairs associations, described in more detail in the next section.

4 Evaluation

We evaluate the proposed system on the task of discovering symptoms associated to different diseases from a collection of over 14M PubMed abstracts published between 2000 and 2022. We assess the performance of the proposed pipeline in terms of Recall, F1 Score, and Precision@ k ($P@k$), with $k \in \{1, 3, 5\}$, defined as the number of relevant items among the top k ranked by the system divided by k . As ground truth data, we consider a subset of the Disease-Symptoms Knowledge Database (Wang et al., 2008) published by Columbia University and available online.³ This dataset contains a list of disease-symptom pairs extracted from textual discharge summaries of patients at New York Presbyterian Hospital. We only considered 54 out of 134 diseases from the ground truth data since for the remaining 80 no symptoms were ever associated – i.e. mentioned in the same sentence – with the respective disease in our subset of PubMed abstracts. After pruning, our ground truth data contained an average of 1.89 symptoms per disease (min=1, max=5, standard deviation=1.10). For this reason, we limit the number of candidate symptoms returned by our approach to a maximum of 10 per disease after ranking them, as explained before. To determine the parameters α and β used to compute the final ranking score of each disease-symptoms candidate, we perform a 2-fold cross validation optimizing the $P@1$ performance measure of the entire pipeline.

Disease-Symptoms Associations Discovery.

From the evaluation results reported in Table 1, we observe that the proposed pipeline employing both the ARM and BRC models is able to rank among the top 10 symptoms all the ones reported in our ground truth data – i.e. achieves a Recall of 1.00. We also observe that, between the ARM and BRC models, the ARM model is superior in

³<https://people.dbmi.columbia.edu/~friedma/Projects/DiseaseSymptomKB>.

	P@1	P@3	P@5	Recall	F1 Score
ARM	0.56	0.33	0.25	0.75	0.29
BRC	0.09	0.10	0.12	0.83	0.21
ARM and BRC	0.57	0.33	0.25	1.00	0.31

Table 1: Performance evaluation of the proposed pipeline employing only the Association Rules Mining component (ARM), the Binary Relation Classification (BRC) one and both of them (ARM and BRC).

detecting significant associations between diseases and their symptoms in our ground truth. On the other hand, by design, the BRC model prefers symptoms which are mentioned a few times in the corpus but have a very strong semantic association with their respective disease. For instance, the BRC model might more easily detect a rare symptom for a disease which is only mentioned a few times in the scientific literature than the ARM approach. Conversely, the ARM approach distinguishes between frequent and infrequent entities and for this reason is better able to capture associations between common symptoms with their respective diseases. For this reason, the BRC model achieves a lower $P@k$ than ARM but a higher recall since the frequency of co-occurrence of disease-symptoms pairs does not affect the relevance estimation process of this component. Thanks to the different characteristics of the two models, the combined approach employing both elements in the pipeline achieves overall a higher $P@k$ and Recall as the two models are able to complement each other. Overall, the lower $P@3$ and $P@5$ values that we observe are motivated by the small number of relevant symptoms for each disease. For example, in most of the cases, a disease has less than 5 recognized symptoms and for this reason its $P@5$ will be lower than 1.0 even if all the correct symptoms have been retrieved by our model.

Qualitative Evaluation. Quoting Wang et al. (2008) – who created the ground truth of disease-symptom associations that we employ in our evaluation – “One of the limitations in this study is that these associations are based on inpatient reports and therefore may reflect different disease-symptom associations than those that would be acquired using reports from outpatients”. For this reason, we include a qualitative assessment of the relevance of the discovered disease-symptoms pairs on our collection of PubMed abstracts. We also use this assessment to show how the sup-

port statements provided by our system to each of the disease-symptom association claims could be used in practice to evaluate and interpret the results of our pipeline. As shown in Table 2, for each of the selected disease-symptom pairs, the proposed pipeline also returns at least one sentence that provides some evidence to support its disease-symptom association claims, this allows us to easily verify whether some of disease-symptom pairs are actually not relevant or are just missing from the ground truth. Using this evidence, we man-

Disease	Symptom	Support Statement	PMID
Asthma	Dyspnea	Dyspnea is a prominent symptom in asthma.	21635136
Bronchitis	Coughing	For example, midnight worsening of cough is a frequent complaint of patients with laryngitis and bronchitis.	18346860
Dyspnea	Deglutition Disorders	Dysphagia in children most commonly presents as feeding or respiratory difficulty.	14992456

Table 2: Sample of disease-symptom pairs discovered by our approach. We also report one of the statements retrieved in support of each association and indicate the PubMed ID (PMID) of the paper reporting it.

ually verified the relevance of each of the top 3 symptoms recognized for each of the diseases by our system. We evaluated the semantic relation of each disease-symptom pair in the top 10 sentences extracted from PubMed abstracts provided as evidence by the model and updated the ground truth if we found any. We share the new disease-symptom associations obtained from this process and the respective PubMed abstracts sentences provided automatically as supplementary material to this paper. As a results of our manual evaluation, we decided to add 70 new disease-symptoms pairs to the ground truth – 1.29 new symptoms for each of the considered diseases. A new disease-symptom association was added to the ground truth if we recognized a statement validating that association among the evidence provided from PubMed by our model. During this process, we observed a few counter-intuitive associations retrieved by proposed pipeline that were semantically correct according to the UMLS semantic network but logically incorrect. For example, we observed 18 associations between different diseases and the UMLS entity “Illness (finding)” which is classified under the semantic type “Sign or Symptom” – e.g. in the sentence “When illness occurs, it is primarily a pneumonic presentation.” stating a relation between the disease “Pneumonia” and the “Illness (finding)” entity. We marked these associations as not relevant in our

ground truth even if we observed a semantic relation between the entities in the provided sentences. Finally, we repeated the evaluation of the proposed pipeline considering the updated ground truth and observed higher values of P@1, P@3 and P@5 of 0.87, 0.77, 0.47, respectively and a Recall of 0.99.

5 Conclusions, Limitations and Future Work

We describe an end-to-end pipeline for the accelerated discovery of medical entity associations from textual data. We evaluate the proposed approach on the task of extracting disease-symptom pairs from medical literature. The main advantages of the proposed system are (i) its capability to discover new entity associations from collections of millions of scientific abstracts, (ii) the ability to easily include new scientific data in its index and therefore to perform up-to-date predictions, (iii) the independence from human annotations, (iv) the interpretability of its predictions given supporting evidence, (v) its low computational complexity, and (vi) the reliance only on open source libraries and models. We believe the adoption of systems like this in the healthcare industry could help medical professionals and researchers in making better-informed decisions. Such systems could also accelerate scientific discovery by giving researchers the ability to quickly verify potential entity associations claims against scientific literature, or discover new symptoms if used with additional data sources. Our pipeline could also be employed to extend existing resources such as the UMLS ontology with new entity relations.

The proposed approach allows researchers to verify the generated entity associations claims by providing statements from indexed scientific documents that motivate such claims. Despite the encouraging results, the quality of the associations discovered is limited by the accuracy of the entity annotator (incorrect annotations observed in recognizing acronyms may lead to inaccurate entity associations). Similarly, the available UMLS semantic types may not accurately describe the different categories and this can also introduce noise. In addition, the system uses only PubMed abstract, while the entire text could provide more information. As future work, we would like test the pipeline over a larger ground truth, explore relations between other concepts and explore the issue of veracity of the extracted claims in cases where the opinions in

scientific literature change over time. We are also planning to evaluate possible improvements to our pipeline, in particular at the stage where the BRC and ARM scores are combined, exploring some machine learning-based approaches for learning to rank.

6 Ethical Considerations

The goal of the proposed system is to support medical professionals and researchers in the accelerated discovery of new entity associations such as new disease-symptom pairs. We show how to do so by relying on the vast collection of medical literature available in PubMed. During the design of the proposed pipeline we paid particular attention to the interpretability and transparency of the results provided by this system. By providing explicit statements in support to each of the discovered associations with references to peer-reviewed scientific publications, we expect users of this system to independently verify the veracity of the provided information and to keep updated the index of publications on which the search system relies. Machine learning models are imperfect and could therefore misinterpret user input or make certain predictions without a having a complete view or understanding of their context. These errors could have serious consequences, especially in the healthcare domain. For this reason, researchers and healthcare professionals employing this system should be aware of possible harms and risks stemming from the use of it, and should implement appropriate safeguards to guarantee the safety of their patients. We also believe that the data we share as supplementary material should not be considered as a verified resource for disease-symptoms associations for any healthcare application as no medical professional reviewed the correctness of our annotations. Finally, since the proposed approach relies on a collection of textual documents for the discovery of new entity associations, it is important that this collection contains an unbiased representation of the entire target population. Otherwise, the system might exhibit a poor performance when interrogated on aspects that might be prevalent in underrepresented populations.

References

- Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference of*

- Very Large Data Bases, VLDB, volume 1215, pages 487–499. Santiago, Chile.
- Alan R. Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association.
- Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.
- Cohen K. Bretonnel and Hunter Lawrence. 2008. Getting started in text mining. *PLoS computational biology*, 4(1):e20.
- Elizabeth S. Chen, George Hripcsak, Hua Xu, Marianthi Markatou, and Carol Friedman. 2008. Research paper: Automated acquisition of disease-drug knowledge from biomedical and clinical documents: An initial study. *Journal of American Medical Informatics Assoc.*, 15(1):87–98.
- Kenneth Church and Patrick Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational linguistics*, 16(1):22–29.
- Berry De Bruijn, Colin Cherry, Svetlana Kiritchenko, Joel D. Martin, and Xiao-Dan 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 : JAMIA*, 18:557 – 562.
- Cyril Grouin, Asma B. Abacha, Delphine Bernhard, Bruno Cartoni, Louise Deléger, Brigitte Grau, Anne-Laure Ligozat, Anne-Lyse Minard, Sophie Rosset, and Pierre Zweigenbaum. 2010. Caramba: Concept, assertion, and relation annotation using machine-learning based approaches.
- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.
- Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. [Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5203–5213, Florence, Italy. Association for Computational Linguistics.
- Patrick Jon D. and Li Min. 2010. High accuracy information extraction of medication information from clinical notes: 2009 i2b2 medication extraction challenge. *Journal of the American Medical Informatics Association : JAMIA*, 17 5:524–7.
- Vinod D. Kumar and Hannah J. Tipney. 2014. *Biomedical literature mining*. Springer.
- Tomohiro Morita, Abidur Rahman, Takanori Hasegawa, Akihiko Ozaki, and Tetsuya Tanimoto. 2017. The potential possibility of symptom checker. *International Journal of Health Policy Management*, pages 615–616.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. Scispace: Fast and robust models for biomedical natural language processing. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327.
- Yoonyoung Park, Natasha Mulligan, Martin Gleize, Morten Kristiansen, and Joao H. Bettencourt-Silva. 2021. Discovering associations between social determinants and health outcomes: Merging knowledge graphs from literature and electronic health data. In *AMIA Annual Symposium Proceedings*, volume 2021, page 940. American Medical Informatics Association.
- Parth Pathak, Pinal Patel, Vishal Panchal, Sagar Soni, Kinjal Dani, Amrish Patel, and Narayan Choudhary. 2015. ezDI: A supervised NLP system for clinical narrative analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 412–416, Denver, Colorado. Association for Computational Linguistics.
- Setu Shah, Xiao Luo, Saravanan Kanakasabai, Ricardo Tuason, and Gregory Klopper. 2019. Neural networks for mining the associations between diseases and symptoms in clinical notes. *Health information science and systems*, 7(1):1–9.
- Alisa Smirnova and Philippe Cudré-Mauroux. 2018. Relation extraction using distant supervision: A survey. *ACM Computing Surveys (CSUR)*, 51(5):1–35.
- Chris Stark, Bobby-Joe Breikreutz, Teresa Reguly, Lorrie Boucher, Ashton Breikreutz, and Mike Tyers. 2006. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539.
- 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.
- Chen Wang, Gang Hu, Kui Wang, Michal Brylinski, Lei Xie, and Lukasz Kurgan. 2016. Pdid: database of molecular-level putative protein–drug interactions in the structural human proteome. *Bioinformatics*, 32(4):579–586.
- Xiaoyan Wang, Amy Chused, Noémie Elhadad, Carol Friedman, and Marianthi Markatou. 2008. Automated knowledge acquisition from clinical narrative reports. In *AMIA Annual Symposium Proceedings*, volume 2008, page 783. American Medical Informatics Association.

Shweta Yadav, Srivastva Ramesh, Sriparna Saha, and Asif Ekbal. 2020. Relation extraction from biomedical and clinical text: Unified multitask learning framework. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

Yijia Zhang and Zhiyong Lu. 2019. Exploring semi-supervised variational autoencoders for biomedical relation extraction. *Methods*, 166:112–119.

A Supplementary Material

In Table 3, we report all the disease-symptoms pairs that we consider as ground truth for our evaluation in Section 4. We expand the list of associations provided by Wang et al. (2008) by manually evaluating the associations between the top 3 symptoms retrieved for each disease by the proposed pipeline. We considered a disease-symptom association as relevant if we could find at least one sentence – among the top 10 provided by our model from our collection of PubMed abstracts – that confirmed a generalized association between each disease and candidate symptom. For each of the pairs we added, we also report the respective statement that motivated our decision. Disease-symptom pairs for which we do not provide a support statement are part of the associations already provided in (Wang et al., 2008).

Disease	Symptom	Support Sentence (when added to original GT)
Anemia	Fatigue	–
Anemia, Sickle Cell	Chronic pain	Chronic pain affects 50% of adults with sickle cell disease (SCD).
Anemia, Sickle Cell	Syncope	Although benign mechanisms predominate, syncope may be arrhythmic and precede SCD.
Anemia, Sickle Cell	Pain	–
Anxiety state	Fatigue	Fatigue was correlated with depression ($r = .40$, $p < .01$), state anxiety ($r = .40$, $p < .01$), and trait anxiety ($r = .46$, $p < .01$).
Anxiety state	Pain	–
Asthma	Wheezing	This report serves as a reminder to all clinicians that "not all that wheezes is asthma".
Asthma	Dyspnea	Dyspnea is a prominent symptom in asthma.
Asthma	Coughing	–
Bronchitis	Coughing	For example, midnight worsening of cough is a frequent complaint of patients with laryngitis and bronchitis.
Bronchitis	Fever	–
Bronchospasm	Dyspnea	–
Cellulitis	Fever	–
Cellulitis	Pain	–
Cholecystitis	Fever	We report a 55-year-old man who presented with fever and abdominal pain compatible with cholecystitis.
Cholecystitis	Vomiting	We present the case of a 69 year old woman with a history of cholecystitis, who consulted for severe abdominal pain, nausea and vomiting.
Cholecystitis	Abdominal Pain	–
Chronic Obstructive Airway Disease	Signs and Symptoms, Respiratory	COPD is characterized by episodic increases in respiratory symptoms, so-called exacerbations.
Chronic Obstructive Airway Disease	Dyspnea	–
Chronic Obstructive Airway Disease	Coughing	–
Confusion	Seizures	All patients presented with fever and disorientation; 6 of the 9 (66%) presented with seizures.
Confusion	Headache	–
Congestive heart failure	Cheyne-Stokes Respiration	Cheyne-Stokes respiration is frequently observed in congestive heart failure.
Congestive heart failure	Angina Pectoris	He often had episodes of angina at night or during dialysis, and then developed congestive heart failure and was hospitalized.
Congestive heart failure	Dyspnea	–
Deep Vein Thrombosis	Syncope	Symptoms significantly associated with DVT were syncope and chest pain.
Deep Vein Thrombosis	Headache	The clinical picture of deep cerebral vein thromboses (DCVT) usually is acute, combining vigilance disorders, headaches, and focal neurologic deficit.
Deep Vein Thrombosis	Pain	–
Degenerative polyarthritis	Pain	Osteoarthritis is clinically defined mainly by pains upon movement and joint stiffness.
Degenerative polyarthritis	Knee pain	Osteoarthritis (OA) is a major source of knee pain.
Deglutition Disorders	Dyspnea	Dysphagia in children most commonly presents as feeding or respiratory difficulty.
Deglutition Disorders	Hoarseness	–
Dehydration	Diarrhea	–
Dehydration	Vomiting	–

Delirium	Agitation	Agitation can be one of the early signs of delirium or altered mental status (AMS).
Delirium	Malaise	Delirium is highly prevalent in critically ill patients.
Delusions	Psychotic symptom	The psychotic symptoms were variable with delusions and/or hallucinations.
Delusions	Agitation	–
Delusions	Hallucinations, Auditory	–
Diabetic Ketoacidosis	Abdominal Pain	Abdominal pain is a frequent manifestation in patients presenting with Diabetic Ketoacidosis (DKA).
Diabetic Ketoacidosis	Vomiting	–
Diverticulitis	Abdominal Pain	Abdominal pain is the most common complaint in patients with acute diverticulitis.
Diverticulitis	Pain	Acute diverticulitis is a painful disease of the colon characterized by peridiverticular inflammation and/or infection.
Diverticulitis	Fever	–
Epilepsy	Seizures	Epilepsy is the most prevalent neurological disease and is characterized by recurrent seizures.
Epilepsy	Fever	Temporal-parietal-occipital carrefour epilepsy is part of the genetic epilepsy with febrile seizures plus spectrum.
Epilepsy	Headache	Epilepsy bears a bidirectional relationship with headache.
Gastritis	Dyspepsia	Gastritis, GERD, and PUD are the leading causes of dyspepsia.
Gastritis	Diarrhea	Gastritis is an inflammatory disease leading to abdominal pain, nausea, and diarrhea.
Gastritis	Abdominal Pain	–
Gastroenteritis	Diarrhea	Gastroenteritis is a common disease in children, characterized by diarrhea, vomiting, abdominal pain, and fever.
Gastroenteritis	Fever	–
Gastroesophageal reflux disease	Chronic cough	Gastro-esophageal reflux can be the cause of chronic cough.
Gastroesophageal reflux disease	Dyspepsia	Gastritis, GERD, and PUD are the leading causes of dyspepsia.
Gastroesophageal reflux disease	Heartburn	–
Gout	Foot pain	Gout is associated with foot pain, impairment, and disability.
Gout	Fever	In conclusion, gout attacks in elderly patients are associated with fever and higher ESR and CRP levels, often resembling a septic arthritis.
Gout	Pain	–
Heart failure	Dyspnea	–
Hemorrhoids	Pain	–
Hemorrhoids	Diarrhea	–
Hepatitis	Icterus	Onset of hepatitis was defined as jaundice and elevated alanine aminotransaminase (ALT) levels.
Hepatitis	Fever	–
Hypothyroidism	Diarrhea	Diarrhoea and malabsorption are common findings together with hyperthyroidism, whereas constipation is frequently observed in hypothyroidism.
Hypothyroidism	Dry skin	Dry skin may be a manifestation of hypothyroidism.
Hypothyroidism	Fatigue	–
Ileus	Vomiting	Many cases of terminal cancer develop ileus symptoms such as vomiting and abdominal distension.
Ileus	Abdominal Pain	–
Ileus	Constipation	–
Influenza	Headache	–
Mental Depression	Pain	The link between pain and depression lies in the central and peripheral nervous systems.
Mental Depression	Fatigue	Depression was the main factor influencing fatigue among both, MS patients and controls.
Mental Depression	Depressive Symptoms	To study the effect of depression (high levels of depressive symptoms) on social engagement.
Migraine Disorders	Headache	Diet can play an important role in the precipitation of headaches in children and adolescents with migraine.
Migraine Disorders	Pain	Migraineurs have atypical pain processing, increased expectations for pain, and hypervigilance for pain.
Migraine Disorders	Vertigo	Migraine is a common cause of vertigo.
Neuropathy	Neuralgia	Neuropathic pain is the most common type of pain in neuropathy.

Neuropathy	Ataxia	JCV granule cell neuronopathy (JCV-GCN) is caused by infection of cerebellar granule cells, causing ataxia.
Neuropathy	Pain	–
Osteomyelitis	Pain	Osteomyelitis ossis pubis is a painful disorder.
Osteomyelitis	Fever	–
Osteoporosis	Weakness	Osteoporosis is a debilitating disease.
Osteoporosis	Perceived quality of life	These indicate that osteoporosis decreased QOL.
Osteoporosis	Pain	–
Pancreatitis	Pain	Pain is a main complaint of patients with pancreatitis.
Pancreatitis	Icterus	Features of pancreatitis were present in 59, cholangitis in 26 and jaundice in 109 patients.
Pancreatitis	Abdominal Pain	–
Pancytopenia	Hepatosplenomegaly	Examination revealed hepatosplenomegaly associated with pancytopenia.
Paranoia	Sleeplessness	Recent epidemiological studies show a strong association of insomnia and paranoia.
Paranoia	Agitation	–
Parkinson Disease	Motor symptoms	Motor symptoms in Parkinson's disease (PD) patients are usually asymmetric at onset.
Parkinson Disease	Bradykinesia	Motor slowness (bradykinesia) is a core feature of Parkinson's disease (PD).
Parkinson Disease	Tremor	–
Peptic Ulcer	Dyspepsia	Gastritis, GERD, and PUD are the leading causes of dyspepsia.
Peptic Ulcer	Pain	This pain is related to extrahepatic infusion and gastroduodenal ulceration.
Pericardial effusion	Fever	The more common symptoms associated with purulent pericardial effusion are fever, dyspnea, and tachycardia.
Pericardial effusion	Chest Pain	Pericardial effusion was diagnosed because the child suffered chest pain and fatigue.
Pericardial effusion	Dyspnea	–
Pneumonia	Fever	–
Pneumothorax	Respiratory distress	Spontaneous pneumothorax is a recognised cause of respiratory distress in the neonatal period.
Pneumothorax	Hemoptysis	–
Psychotic Disorders	Seizures	Out of these 8 patients, 3 presented with psychosis (12.5%) and 4 (17%) with seizures.
Pulmonary Edema	Respiratory distress	The respiratory distress was initially caused by pulmonary edema and later was caused by severe bronchorrhea.
Pulmonary Edema	Dyspnea	–
Pulmonary Embolism	Syncope	Syncope can be caused by a pulmonary embolism.
Pulmonary Embolism	Dyspnea	–
Pulmonary Embolism	Chest Pain	–
Pyelonephritis	Flank Pain	Symptoms of cystitis are dysuria, frequency, new onset incontinence and malodorous urine while symptoms of pyelonephritis are high grade fever, flank pain and vomiting.
Pyelonephritis	Dysuria	We report the case of a 45-year-old African American man who presented with symptoms of right-sided pyelonephritis, including fever, dysuria, and flank pain.
Pyelonephritis	Fever	–
Respiratory Failure	Dyspnea	–
Thrombocytopenia	Fever	–
Thrombocytopenia	Fatigue	–
Tonic-Clonic Epilepsy	Seizures	Seizures are reported in one quarter, including tonic-clonic, absence, and febrile seizures.
Tonic-Clonic Epilepsy	Fever	Onset in the first year of life by febrile or afebrile clonic and tonic-clonic, generalized, and unilateral seizures, often prolonged, in an apparently normal infant is the first symptom, suggesting the diagnosis.
Tonic-Clonic Epilepsy	Myoclonus	–
Transient Ischemic Attack	Neurologic Symptoms	Transient ischemic attack (TIA) is a cerebrovascular disease with temporary (<24 h) neurological symptoms.
Transient Ischemic Attack	Seizures	Little attention has been paid to the possibility that seizures may be precipitated by TIAs.
Transient Ischemic Attack	Headache	–
Upper Respiratory Infections	Coughing	The commonest form of cough is caused by upper respiratory tract infection and has no benefit to the host.

Upper Respiratory Infections	Fever	-
Urinary tract infection	Fever	Patients' clinical status was dominated by fever due to upper urinary tract infection.
Urinary tract infection	Dysuria	-

Table 3: Extended list of disease-symptoms associations. We provide a supporting statement for each of the associations that we decided to add to the list of disease-symptom associations provided by [Wang et al. \(2008\)](#).

PENTATRON: Personalized coNText-Aware Transformer for Retrieval-based cOnversational uNderstanding

Niranjana Uma Naresh* Ziyang Jiang* Ankit*
Sungjin Lee Jie Hao Xing Fan Chenlei Guo

Amazon

{niumanar, ziyjiang, ankitvys, sungjinl, jieha, fanxing, guochenl}@amazon.com

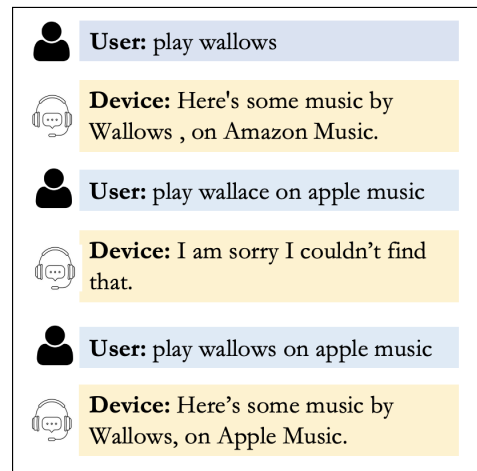
Abstract

Conversational understanding is an integral part of modern intelligent devices. In a large fraction of the global traffic from people using smart digital assistants, frictions in dialogues may be attributed to incorrect understanding of the entities in a user’s query due to factors including ambiguous mentions, mispronunciation, background noise and faulty on-device signal processing. Such errors are compounded by two common deficiencies from intelligent devices namely, (1) the device not being tailored to individual users, and (2) the device responses being unaware of the context in the conversation session. Viewing this problem via the lens of retrieval-based search engines, we build and evaluate a scalable entity correction system, PENTATRON. The system leverages a parametric transformer-based language model to learn patterns from in-session user-device interactions coupled with a non-parametric personalized entity index to compute the correct query, which aids downstream components in reasoning about the best response. In addition to establishing baselines and demonstrating the value of personalized and context-aware systems, we use multitasking to learn the domain of the correct entity. We also investigate the utility of language model prompts. Through extensive experiments, we show a significant upward movement of the key metric (Exact Match) by up to 500.97% (relative to the baseline).

1 Introduction

Intelligent devices are ubiquitous in the modern computing. The scientific modules that drive these devices involve conversational understanding, ambient computing, natural language reasoning and self-learning (Thoppilan et al., 2022; Sarikaya, 2022; Pinhanez et al., 2021; Liu et al., 2021). A user’s interaction with a device, however, is susceptible to errors arising from a myriad of sources including wrong pronunciation, inaccuracies in

the subject mentions in a sentence, environmental noise, hardware and software error (Kim et al., 2020). Correct interpretations of user queries, especially entities, is central to delivering the best user experience. Two important factors that contribute strongly to high-precision entity recognition are (1) personalization, ie, learning users’ unique patterns, and (2) contextualization, ie, deriving cues from the information in a user-device interaction session. In this paper, we design and evaluate an entity correction system, PENTATRON, with both personalization and contextualization baked into its architecture.



(a)

“[USER] play wallows [DEVICE] Here's some music by Wallows , on Amazon Music. [USER] play wallace on apple music”

(b)

Figure 1: (Above) One multi-turn dialogue session with defective source query which contains one erroneous entity ‘wallace’ and its successful rephrase with correct entity ‘wallows’. (Below) Concatenation of queries and responses using special tokens to form a single sequence as encoder input.

*Equal contribution.

1.1 Motivation

In Figure 1, we illustrate a real-world case as to why personalization and contextualization are very important, especially due to the specificity in highly entity-centric domains such as music. In this case, masking the very last device response, we observe that there is valuable information scattered across the user’s requests in the session yet, the device delivers sub-par experience by responding defectively multiple times before finally getting the user’s intent right.

1.2 Notation and Preliminaries

Definition 1. Let integer γ satisfy $1 \leq \gamma < \infty$. A natural language (NL) hypothesis is a mapping, $h : Q \rightarrow D \times I \times [E]^\gamma$, where Q refers to the query space, D refers to the domain space, I refers to the intent space and E refers to the entity space. The entity space, $E := E_T \times E_V$, may further be decomposed into the entity type space E_T and the entity value space E_V . All spaces are defined over Unicode strings.

As an example, given a query string $q = \text{“play the real slim shady”}$, the corresponding NL hypothesis is $h(q) = (\text{Music}, \text{PlayMusicIntent}, [(\text{SongName}, \text{the real slim shady})])$ where the domain is *Music*, the intent is *PlayMusicIntent*, and the entity value is *the real slim shady* with *SongName* entity type.

Definition 2. Building on Definition 1, our system, PENTATRON, may be formalized as $\Phi : (C, Q) \rightarrow E_V$ where C is the user space (anonymized using a hash function, for privacy, in practice).

In a nutshell, given an input query q (with or without dialogue context), our system essentially solves the optimization problem,

$$\min_{\theta} \mathbb{E}_{(c,q,e) \sim \mathcal{D}} [\ell(\Phi_{\theta}(c, q), e)] \quad (1)$$

where \mathcal{D} is supported on $C \times Q \times E_V$.

1.3 Our Contributions and Preview of Results

On the system design front, we build a retrieval-based pipeline. Our model backbone is inspired by attention-based (Vaswani et al., 2017) transformer encoders (Devlin et al., 2018). We achieve personalization via a non-parametric index which is essentially a key-value pair look-up table with the keys representing users and values representing the entity lists derived from historical data aggregation. With respect to experimental results, we

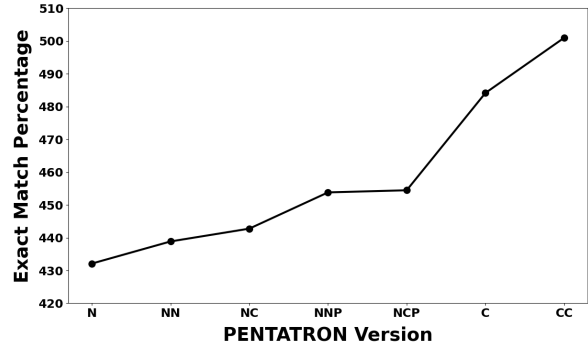


Figure 2: Preview of the system performance which shows consistent significant improvement in going from a purely personalized system (N) to a fully contextual personalized system (CC). Further details are available in Table 1.

conduct extensive studies on seven different versions of PENTATRON, involving ablations with prompts, multi-tasking and non-contextual training data, and show consistent improvements in Exact Match (EM) of up to 500.97% (relative to the baseline) as captured by the preview of results in Figure 2.

2 Background and Related Work

2.1 Query Rewriting

Query Rewriting (QR) in dialogue systems aims to reduce frictions by reformulating the automatic speech recognition component’s interpretation of users’ queries. Initial efforts (Dehghani et al., 2017; Su et al., 2019) treat QR as a text generation problem.

Some recent studies (Chen et al., 2020; Yuan et al., 2021; Fan et al., 2021; Cho et al., 2021) are based on neural retrieval systems. In the retrieval-based systems, the rewrite candidate pool is aggregated from users’ habitual or historical queries so that the rewrite quality can be tightly controlled. Compared to generation-based systems, retrieval-based systems may sacrifice flexibility and diversity of the rewrites, but in the meanwhile provide more stability which is more important in a runtime production setup.

Personalization and **Contextualization** are two popular directions for QR systems. A personalized system such as Cho et al., 2021 tends to incorporate diverse affinities and personal preferences to provide individually tailored user experience in a single unified system. Contextualization attempts to utilize multi-turn queries rather than only leveraging single-turn information. Some previous stud-

ies (Wang et al., 2021) have shown the benefits by leveraging the dialogue context and user-device interaction signals.

Entities have been shown to be a strong indicator of text semantics. Since queries in our dialogue system are typically short sentences, entities are even more important in this scenario. Most existing QR approaches mentioned above rephrase query utterances entirely. Although some existing works focus on specific categories like coreference resolution or entity omission (Su et al., 2019; Tseng et al., 2021), none of them has a particular emphasis on the correction of erroneous entities.

2.2 Entity Linking

Another related thread towards our task is entity linking. Entity linking task aims to link mentioned entities with their corresponding entities in a knowledge base. In a retrieval-based QR system which focuses on entity correction, we could adopt similar methods in entity linking area. BLINK(Wu et al., 2019) designs a two-stage retrieve-rerank framework based on pre-trained deep transformers. The following work ELQ (Li et al., 2020) uses a biencoder to jointly perform mention detection and linking in one pass and also shows good improvement in latency metrics which is quite important in a production settings. Our task is more challenging than entity linking because the input utterance is noisy with incorrect entities and the lack of textual descriptions of each entity.

3 Problem Setup and Solution Design

The overall architecture of the PENTATRON system is described in Figure 3.

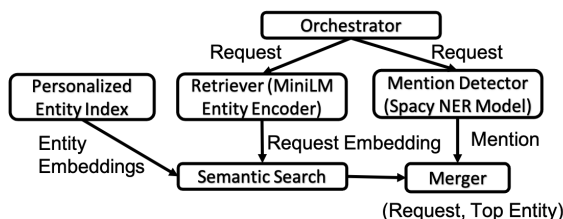


Figure 3: For a given user, the input request string from the PENTATRON orchestrator is processed by a transformer model and also by a named entity recognition model, both trained on historical user requests, to encode the request and extract mentions, respectively. A semantic search is applied on the request embeddings and the precomputed entity embeddings of the user to find the best match following which, post-processing is applied to feed the result into downstream components.

3.1 Entity Correction in Query Rewriting

We consider a dataset of M multi-turn dialogue sessions: $\{S_t\}_{t=1}^T$. S is a set of T turns in chronological order: $S = \{(q_t, r_t)\}_{t=1}^T$. Here t is the index of turn and each turn consists of a pair (q_t, r_t) , where q_t denotes the user’s query utterance and r_t denotes the device’s response utterance. The sessions are selected so that the source query q_{T-1} contains one erroneous entity and q_T , which has the correct form of entity e , is the rephrase of the previous turn. More details about the data selection is described in Section 4.1. Our prediction goal is formulated as:

$$\hat{e} = \operatorname{argmax}_e \mathbf{P}(e \mid \{S_t\}_{t=1}^{T-2}, q_{T-1}) \quad (2)$$

$$\hat{q}_T = g(q_{T-1}, \hat{e}) \quad (3)$$

We flatten the previous dialogue turns $\{S_t\}_1^{T-2}$ and the source query q_{T-1} into a single sequence to feed into the encoder, as shown in Figure 1. Since the only difference between q_{T-1} and q_T is whether we have the correct form of e , the final rewrite is generated based on source query q_{T-1} and entity prediction \hat{e} through a simple replacement function g .

3.2 Personalized Entity Index

We build an personalized entity index for each user to leverage individual interaction history by aggregating users’ frequent entities in past 30 days¹. The entities include song names or artists that users frequently listened to, nicknames of users’ intelligent devices and so on.

This index serves as the retrieval candidate pool during inference time. The candidate embeddings are cached. We implement a two-stage in-memory index that has a map of users to their specific entities along with the embeddings corresponding to the union of entities across all users. This is done for memory efficiency reasons so that we avoid the overhead caused by the redundancy of duplicate entities across different users.

3.3 Modeling

We use a bi-encoder architecture based on MiniLM (Wang et al., 2020) for jointly encoding the queries and the entities (Humeau et al., 2019). The weights are shared for memory footprint savings and serving cost reduction. Note, we also try asymmetric query and candidate entity encoders;

¹All user information is in a de-identified format.

however, we observe only a marginal performance improvement of less than 1%. We use a batch size of 128 and train it on p3.2x-large GPU instances acquired on AWS cloud. AdamW (Loshchilov and Hutter, 2017) is our optimizer of choice.

For detecting mentions in the input query, we use a Spacy named entity recognition model trained on historical user queries containing entity strings from different domains.

3.4 Optimization Objectives

A combination of both hard negatives (Gillick et al., 2019) and in-batch random negatives improve the performance of large-scale natural language reasoning systems. We use the multiple negatives ranking loss (Henderson et al., 2017) for the primary task. We take a metric learning approach (Hadsell et al., 2006) to the auxiliary task, ie, we use the contrastive loss here.

Inference: The semantic search function which is used in primary retrieval task computes $s_i = \cos(f(q), f(e_i))$ for $i \in [k]$ where $e_i \in E_V$ are the top- k entities retrieved from the personalized index (sorted by the relevance score in descending order) and q refers to the query (with or without context). We configure our system to be activated on the threshold conditions, $s_1 > \tau_1$ and $s_2 < \tau_2$, to make sure the top-2 entities are sufficiently far apart to avoid any ambiguous predictions.

Training: The encoder model of the PENTATRON system is trained with the primary task of entity prediction, which we maximize the similarity score between the user query (with or without context) and the target correct entity. Consider a batch of N samples. The loss of the primary task is given by:

$$\mathcal{L}_E = -\frac{1}{N} \sum_{i=1}^N \log \frac{\exp(s_i)}{\sum_{j=1}^N \exp(s_j)} \quad (4)$$

In the above formula, we only take in-batch random negatives into consideration. We will also discuss the utilization of hard negatives later in this paper.

We adopt an auxiliary task during training to have an implicit clustering effect of the query embeddings based on target domain. For this task, we want to push source queries targeting to the same domain close to each other and source queries targeting to the different domain away from each other.

For N randomly selected pairs of queries (indexed by i and j) from a batch, the loss of the auxiliary task is the contrastive loss given by:

$$\begin{aligned} \mathcal{L}_D = & \frac{1}{N} \sum_{(i,j)} \mathbf{1}\{h^D(q_i) = h^D(q_j)\} \\ & \|f(q_i) - f(q_j)\|^2 \\ & + \frac{1}{N} \sum_{(i,j)} \mathbf{1}\{h^D(q_i) \neq h^D(q_j)\} \\ & \max(0, \lambda - \|f(q_i) - f(q_j)\|^2) \end{aligned} \quad (5)$$

The margin parameter λ is set as 0.75. Here, h^D denotes the domain extracted from the NL hypothesis of the target (final) dialogue turn.

Multi-task Formulation: The final loss is computed as $\mu\mathcal{L}_E + (1 - \mu)\mathcal{L}_D$ where $\mu \in (0, 1]$. Specifically, we build different versions of PENTATRON by setting $\mu = 1$ and $\mu = 0.5$.

We train two single-task models which are used as the non-contextual and contextual baselines respectively. The non-contextual baseline model uses the source query as input and the rewrite entity as output. The contextual baseline model uses the full context (truncated to maximum allowable length of 256) as input and the rewrite entity as output.

Furthermore, we train another five versions PENTATRON with multi-task settings. We also investigate the usage of task markers similar to the approach in Maillard et al., 2021. The (hard) prompts are added as special tokens [REWRITE] and [DOMAIN] before the corresponding input during training. More details are presented in Table 1

Hard Negatives Mining: First, we use bm25 (Robertson et al., 2009) to mine hard negatives from the candidate pool, which shows minor improvement. Hence, we adopt a two-pass method to compute hard negatives. In the first pass, we use a model trained with random negatives to perform inference on a disjoint ‘‘second’’ training set to obtain entity predictions. In the second pass, we continue training the previous baseline model checkpoint and take into account the wrong predictions as hard negatives.

4 Experiments

4.1 Training and Test Data

Our data is derived from the logs of a commercial voice assistant and we process the data with strict privacy standards so that users are not identifiable.

Model	Primary Task	Auxiliary Task	Exact Match (Relative)
DPR-EC	Non-contextual	None	0.0 [Baseline]
PENTATRON-N	Non-contextual	None	+432.11%
PENTATRON-NN	Non-contextual	Non-contextual	+438.86%
PENTATRON-NC	Non-contextual	Contextual	+442.76%
PENTATRON-NNP	Non-contextual	Non-contextual with prompt	+453.82%
PENTATRON-NCP	Non-contextual	Contextual with prompt	+454.47%
PENTATRON-C	Contextual	None	+484.14%
PENTATRON-CC	Contextual	Contextual	+500.97%

Table 1: List of all model settings and their performance numbers (relative, with respect to the baseline, DPR-EC). The primary task is entity prediction using the multiple negative ranking loss with a batch size of 128 and the auxiliary task uses the online contrastive loss with a margin of 0.75. We apply the state-of-the-art retrieval model, DPR (Karpukhin et al., 2020), to train a dual BERT architecture, DPR-EC, for entity correction as the baseline, i.e., without utilizing personal and contextual information.

We sample multi-turn dialogue sessions between English-speaking users and devices in a time period of one month, in May-June 2022, from all over the United States. A defect detection model similar to (Gupta et al., 2021) and rule-based filters are applied to find dialogue sessions whose last two turns of user query are rephrase pairs. Rule-based filters are using edit-distance and time gap between utterance pairs similar to (Cho et al., 2021). Since our work has a particular emphasis on the correction of erroneous entities, we also utilize the NL hypothesis of the rephrase pairs to get such cases. For simplicity, we consider data with only a single erroneous entity as the target to be predicted. It is straightforward to generalize our system to the multiple entities case.

We sample the test set and keep only sessions wherein a retrieval-based system such as (Cho et al., 2021), which rephrases query utterances entirely, couldn’t solve. For training, a sample of two million utterances was extracted. Also, some (completely generic) example dialogs extracted from critical data are reported in the paper (Table 2).

Figures 4 and 5 summarize the keys data statistics on the training and test sets. This gives us an insight into how transformer models stand to benefit from longer sequences in our application since they are parameterized by and compute second-order statistics.

4.2 Evaluation Metrics

We utilize the harshest metric to evaluate our system namely, the Exact Match (EM). This score is 1 if the predicted rewrite exactly matches the labeled rephrase, and is 0 otherwise. We use the same

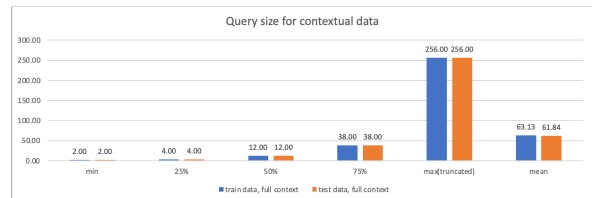


Figure 4: Query length statistics of contextual training and test data.

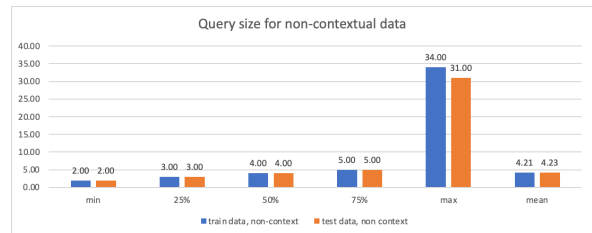


Figure 5: Query length statistics of non-contextual training and test data.

threshold τ_1 and τ_2 for all the proposed PENTATRON models. The threshold is experimentally set up to keep the balance between opportunities and potential risks in real production.

4.3 Observations and Case-studies

Table 1 shows the main results of our different versions of systems. The experimental result is consistent with our intuition. Since the pipeline has also actually been run on live traffic, through an A/B experiment (section 4.4), the baselines were created for the purpose of this paper. All the PENTATRON models benefit from a personalization settings and outperform a global-wise retrieval model DPR-EC by a large margin. Among different settings of PENTATRON, it’s obvious that both contextualiza-

Dialogue Context	[USER] Turn on ben’s light. [DEVICE] I’m sorry I couldn’t find the device. [USER] Turn on benny’s light. [DEVICE] Okay.	[USER] Play calen playlist. [DEVICE] I could not find that on Amazon Music. [USER] Play scars. [DEVICE] Here’s Scars , by James Bay , on Amazon Music.
User Query	Turn ben’s light on pink	Play playlist karen
Rewrite Label	Turn benny’s light on pink	Play playlist calen
DPR-EC	Turn brecken’s light on pink ✗	Play playlist cameron ✗
PENTATRON-N	Turn britney’s light on pink ✗	Play playlist carrie ✗
PENTATRON-C	Turn benny’s light on pink ✓	Play playlist carrie ✗
PENTATRON-CC	Turn benny’s light on pink ✓	Play playlist calen ✓

Table 2: Two examples to showcase the importance of full contextualization and personalization.

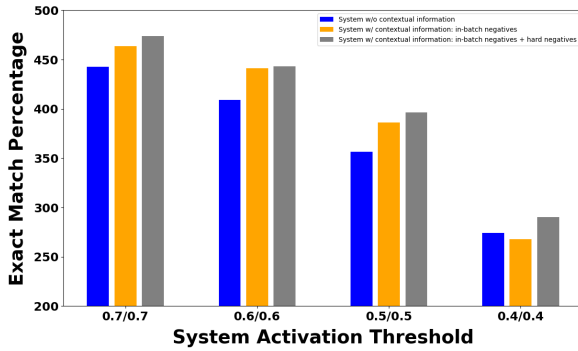


Figure 6: In this figure, we illustrate the importance of contextual information and training with hard negatives in boosting the performance of our system.

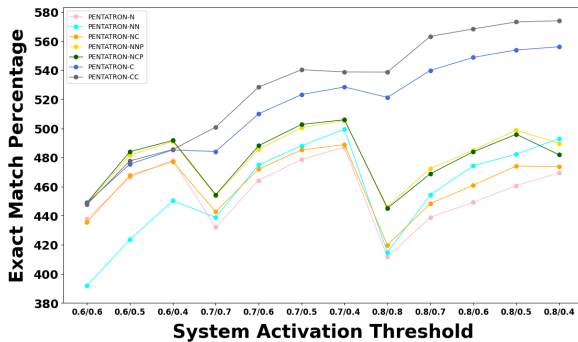


Figure 7: Performance of different versions of PENTATRON with respect to different system activation thresholds τ_1 and τ_2 .

tion and multi-tasking bring further improvement. There is also some gain by adding task markers in the multi-task settings.

Figure 6 presents an ablation that shows the benefits of hard negative sampling. To further stress test our system, we also swept over different thresholds, summarized in Figure 7. We could notice that the general trend is consistent using different thresholds.

In sweeping across thresholds in our empirical studies (Figure 8), we observe interesting trends. In particular, that when $\tau_1 = \tau_2$, the personalized model that does not utilize contextual information suffers from noisy predictions when the thresholds

are equal since the top-2 retrieved entities are semantically very similar and the model finds it difficult to disambiguate. However, with the contextual information, we see consistent improvements in accuracy as we tighten thresholds.

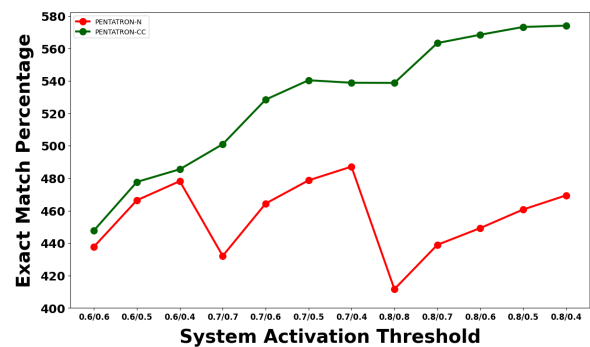


Figure 8: Demonstrating the value of contextual information with appropriate multitasking design.

We illustrate the benefits of our approach on a generic dialog in Table 3. In the left example from *HomeAutomation* domain, the device name in the source query is incorrect which will make this task-oriented dialogue system fail. PENTATRON-C and PENTATRON-CC could generate the correct rewrite by leveraging dialogue context and user’s personalized index which contains user’s registered device name. A similar trend can be observed in the right example from Music domain. Besides, the right example also illustrates the benefits from multi-task learning by comparing the prediction from PENTATRON-C and PENTATRON-CC. Both the video name ‘carrie’ and playlist name ‘calen’ exist in user’s personalized index. With the help of contrastive representation learning, PENTATRON-CC could learn to retrieve a Music domain entity which is the correct one here.

Visualization: We analyze the benefits of our design using t-SNE (Van der Maaten and Hinton, 2008). The results are presented in Figures 9 and 10. We clearly observe that multi-tasking enables domain disambiguation via implicitly clustering the

queries by domains, thus contributing positively to entity prediction accuracy and, in turn, improving the query rewrite quality. In particular, we observe that Music, Video and Knowledge domains immensely benefit from multi-tasking.



Figure 9: In the absence of the auxiliary task, queries across domains are interspersed which leads to lower accuracy due to ambiguity in the rewrite domain. Here, the blue cluster denotes Knowledge domain queries, the orange cluster denotes Music domain queries and the green cluster denotes Video domain queries.



Figure 10: Multi-tasking to predict the rewrite domain, in addition to predicting the correct entity, leads to higher accuracy due to domain disambiguation arising from the implicit clustering effect.

4.4 Online Performance

A/B Experimentation: At the time of writing this, we deployed a static (request, rewrite) look-up table computed using PENTATRON-N to serve real users. With a p -value < 0.05 , we observe a significant improvement, of 47.5%, in the user experience measured using the model-based (Gupta et al., 2021) assessment used for dataset selection in Section 4.1 on the treatment group as compared to the control group. Moreover, other friction metrics such as the turn error rate have improved over 40%

throughout the A/B duration. Successive version upgrade deployments are ongoing.

Latency: To investigate the deployment in a real-time inference service, we performed extensive load tests implemented with a Flask endpoint. We store all objects in the main memory. On a c5.9x-large instance on AWS cloud, at 120 queries per second hitting the PENTATRON system, we observed a P90 latency of less than 30ms for the end-to-end execution.

5 Conclusions and Future Directions

In this work, we build a system called PENTATRON which significantly improves user experience in intelligent devices by operating on entities and reducing friction in multi-turn dialogues. There are several future directions we plan to work on, including operationalizing large-scale unbiased personalized and context-aware systems, and designing self-learning (Ponnusamy et al., 2020; Roshan-Ghias et al., 2020) using techniques such as reinforcement learning. We also plan to investigate the utility of a multi-level index to improve entity coverage and mitigate the cold-start problem for new customers. Dynamic index building and deployment in low-latency applications is an ongoing direction.

Limitations

Our system has the following limitations. Though personalization offers great benefits, the coverage of desired entities in our historical index due to personalization is typically limited. Specifically, we observe only 20% coverage in our empirical studies. This can be alleviated using a multi-level index involving clusters of users. We have initial results on this approach and plan to compile that in future work.

Next, natural language based prompts should further improve our system. However, very long sequence length has concerns with respect to latency and memory on CPU-deployed solutions. A potential solution to this is to consider low-rank factorization in the attention design.

Finally, in production deployments, large-scale in-memory index for multiple locales poses cost challenges. A separate study is warranted to study hybrid storage mechanisms and high performance cache design.

Ethics Statement

To the best of our knowledge, our work is ethical and has a positive impact on society and human well-being. In particular, we take pride in emphasizing that we handle customer confidentiality and privacy with critical care. Its design principles are unbiased.

References

- Zheng Chen, Xing Fan, Yuan Ling, Lambert Mathias, and Chenlei Guo. 2020. [Pre-training for query rewriting in a spoken language understanding system](#). *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*.
- Eunah Cho, Ziyang Jiang, Jie Hao, Zheng Chen, Saurabh Gupta, Xing Fan, and Chenlei Guo. 2021. Personalized search-based query rewrite system for conversational ai. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 179–188.
- Mostafa Dehghani, Sascha Rothe, Enrique Alfonseca, and Pascal Fleury. 2017. Learning to attend, copy, and generate for session-based query suggestion. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1747–1756.
- 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*.
- Xing Fan, Eunah Cho, Xiaojiang Huang, and Chenlei Guo. 2021. Search based self-learning query rewrite system in conversational ai. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldrige, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*.
- Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei Guo. 2021. Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- 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.
- Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. 2017. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*.
- Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969*.
- Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.
- Chanwoo Kim, Dhananjaya Gowda, Dongsoo Lee, Jiyeon Kim, Ankur Kumar, Sungsoo Kim, Abhinav Garg, and Changwoo Han. 2020. A review of on-device fully neural end-to-end automatic speech recognition algorithms. In *2020 54th Asilomar Conference on Signals, Systems, and Computers*, pages 277–283. IEEE.
- Belinda Z Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. *arXiv preprint arXiv:2010.02413*.
- Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model for web-scale retrieval in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3365–3375.
- Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oğuz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-task retrieval for knowledge-intensive tasks. *arXiv preprint arXiv:2101.00117*.
- Claudio Pinhanez, Paulo Cavalin, Victor Henrique Alves Ribeiro, Ana Appel, Heloisa Candello, Julio Nogima, Mauro Pichiliani, Melina Guerra, Maira de Bayser, Gabriel Malfatti, et al. 2021. Using meta-knowledge mined from identifiers to improve intent recognition in conversational systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7014–7027.
- Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya. 2020. Feedback-based self-learning in large-scale conversational ai agents. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13180–13187.
- Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

- Alireza Roshan-Ghias, Clint Solomon Mathialagan, Pragaash Ponnusamy, Lambert Mathias, and Chenlei Guo. 2020. Personalized query rewriting in conversational ai agents. *arXiv preprint arXiv:2011.04748*.
- Ruhi Sarikaya. 2022. Intelligent conversational agents for ambient computing. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 5–5.
- Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance rewriter. *arXiv preprint arXiv:1906.07004*.
- Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.
- Bo-Hsiang Tseng, Shruti Bhargava, Jiarui Lu, Joel Ruben Antony Moniz, Dhivya Piraviperumal, Lin Li, and Hong Yu. 2021. Cread: Combined resolution of ellipses and anaphora in dialogues. *arXiv preprint arXiv:2105.09914*.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(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. *Advances in neural information processing systems*, 30.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.
- Zhuoyi Wang, Saurabh Gupta, Jie Hao, Xing Fan, Dingcheng Li, Alexander Hanbo Li, and Chenlei Guo. 2021. [Contextual rephrase detection for reducing friction in dialogue systems](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1899–1905, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2019. Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*.
- Siyang Yuan, Saurabh Gupta, Xing Fan, Derek Liu, Yang Liu, and Chenlei Guo. 2021. [Graph enhanced query rewriting for spoken language understanding system](#). *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*.

Machine Translation Impact in E-commerce Multilingual Search

Bryan Hang Zhang

Amazon.com

bryzhang@amazon.com

Amita Misra

Amazon.com

misrami@amazon.com

Abstract

Previous work suggests that performance of cross-lingual information retrieval correlates highly with the quality of Machine Translation. However, there may be a threshold beyond which improving query translation quality yields little or no benefit to further improve the retrieval performance. This threshold may depend upon multiple factors including the source and target languages, the existing MT system quality and the search pipeline. In order to identify the benefit of improving an MT system for a given search pipeline, we investigate the sensitivity of retrieval quality to the presence of different levels of MT quality using experimental datasets collected from actual traffic. We systematically improve the performance of our MT systems quality on language pairs as measured by MT evaluation metrics including Bleu and Chrf to determine their impact on search precision metrics and extract signals that help to guide the improvement strategies. Using this information we develop techniques to compare query translations for multiple language pairs and identify the most promising language pairs to invest and improve.

1 Introduction

Multilingual search capability is essential for modern e-commerce product discovery (Lowndes and Vasudevan, 2021; Zhang, 2022). Localization of e-commerce sites have led users to expect search engines to handle multilingual queries. Recent proposals of cross-lingual information retrieval handle multilingual queries, and language-agnostic cross-borders product indexing has gained traction with neural search engines (Hui et al., 2017; McDonald et al., 2018; Nigam et al., 2019a; Lu et al., 2021; Li et al., 2021), but legacy e-commerce search indices are still built on monolingual product information and support for multilingual search is bridged using Query translation (Nie, 2010; Rücklé et al., 2019; Saleh and Pecina, 2020; Bi et al., 2020; Jiang et al., 2020; Zhang and Tan, 2021).

Query translation allows users to look up information represented in documents written in a languages different from the language of the query. It takes as input the query typed in source or query language and returns a translated query to the search engine to retrieve documents in the target language. It follows that query translation plays a key role and its output significantly affects the retrieval results.

Previous studies have demonstrated performance of CLIR (Cross-Lingual Information Retrieval) correlates highly with the quality of the Machine Translation (MT), and improving the quality of MT improves retrieval quality (Goldfarb et al., 2019; Brynjolfsson et al., 2019). However, these evaluations are done separately for each task. This leaves a large gap in understanding the impact of improving MT quality iteratively on CLIR performance in a real time industrial setting. Since machine translation is used here as interim application, the objectives of the retrieval task may have varying levels of tolerance to the inherent translation quality. Information retrieval evaluation usually involves human-annotated relevance labels of search results candidates. In an industry setting, annotating a representative sample is a time consuming and expensive task, particularly during iterative improvement of MT for the search use case. Additionally, a general-purpose MT evaluation metric may not necessarily adapt to the query evaluation for downstream retrieval task.

To address these above concerns, we propose an MT evaluation framework to build an e-commerce specific CLIR test set. It exploits behavioural signals from search retrieval results to evaluate MT quality for a given query. In order to identify the benefit of improving an MT system, we further investigate the sensitivity of retrieval quality to the presence of different levels of MT quality as measured by Bleu, and Chrf using experimental datasets collected from actual traffic. Based on

these experiments, we recommend the pairs that are worth continued investment in improving MT systems for search. Our main contributions are:

- A **rank-based evaluation framework** to evaluate MT in CLIR through ranking-based search metrics using behavioral signals (from the store of the target language) as a proxy to relevance information without any human annotation; this framework can be used to create e-commerce CLIR test set at scale.
- A **method to measure the MT launching impact** on the e-commerce CLIR ecosystems for a given language pair. This can be used to identify and prioritize the high impact language pairs for more investment in the MT improvement.
- A **method to measure the MT improvement impact** on the e-commerce CLIR ecosystems for a given language pair. It signals the strategy to be used for MT improvement, either a comprehensive strategy focusing on the overall query traffic or a specific one targeting a smaller percentage of query traffic or a combination of both strategies.

This paper is organized as following: we propose a rank-based evaluation framework in section 2. We propose two MT impact rates, MT launching impact rate and MT improvement MT rate respectively in section 3. Section 4 is the experiment with 12 language pairs from 6 stores. Section 5 is the results and analysis. We defer related work to Section 6 where we compare it with our proposed work. We draw a conclusion in Section 7.

2 Cross-Lingual Information Retrieval (CLIR) Evaluation Framework for E-commerce Product Search

Different from static test sets in academia, industrial search applications are dynamic as user queries and behavioral signals change with world trends. Moreover, product inventory is dynamic, changes often and quickly.

A previous study (Sloto et al., 2018) proposes the traditional Normalized Discounted Cumulative Gain (nDCG) for CLIR using all search results from the reference translation as relevance ground truth to compute nDCG for MT translation (aka nDCG-MT). However, their approach imposes a strong assumption that the top- k search results

from reference translation are all relevant to the query and relevance is inversely scaled by the ranking of the results.

Although behavioral signals from users’ clicks and purchases are useful proxy (Wu et al., 2018) to expensive human relevance annotations, these are dynamic and change according to the product life cycle and seasonal business trends. These behavioral signals need to be updated at regular cadence to accurately represent relevance information needed to compute search metrics.

We introduce a ranking-based evaluation framework through search ranking metrics using behavioral signals as a proxy to relevance information without any human annotation; To the best of our knowledge, there is no systematic study on cross-lingual information retrieval for e-commerce search that neither requires ground-truth click/purchase information nor human annotated relevance data.

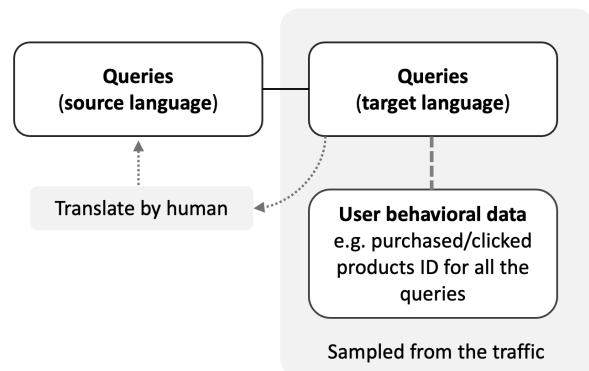


Figure 1: Test set creation workflow

Figure 1 illustrates the test sets creation workflow for MT evaluation in E-commerce CLIR:

1. Create a sample of query data from the historical search traffic in the target language (the language that the search index is built on). Empirically, we recommend to sample that queries from the top 30%, bottom 30% and the middle 40% in frequency bins to better simulate the user traffic. We refer to these queries as Q_{ref} .
2. To allow computation of traditional relevance metrics, aggregate the clicks and/or purchase product IDs associated with the queries, if they are available. We refer to the products IDs associated with the query and their click/purchase frequency as P_{id} and P_{freq} .
3. Create human reference translation of the

search queries sample in the source language (the language that users will be searching in). We refer to these human translated queries as Q_{src} .

We propose the following evaluation framework with the test sets created above to evaluate machine translation in the context of CLIR for e-commerce queries.

1. Translate the Q_{src} with the MT model in consideration. We refer to these machine translated queries as Q_{mt} .
2. Search for the candidate products using the machine translated queries Q_{mt} ; retrieving top- k search result R_{mt}
3. Use P_{id} and P_{freq} (as ground truth) with R_{mt} to compute traditional relevance based metrics such as nDCG.

3 MT impact in the search ecosystem

3.1 The range of MT impact on search

As mentioned, the downstream search pipeline consists of a large number of components, which altogether has different levels of tolerance for query translation quality. Hence, it is important to estimate the range of query translation impact on the search ecosystem in consideration. With the test sets from the creation workflow in Section 2, we propose to measure the rank-based search metrics such as nDCG of source queries Q_{src} as the lower bound of the MT impact, which serves a baseline for the impact of MT translated query on search, and measure the search metrics of human reference query translations Q_{ref} as the upper bound.

3.2 MT launching impact measurement

We expect that launching an MT system in a search ecosystem of different language pairs can have different levels of positive impact on the search result quality. Therefore, given a language pair (e.g. *enus-jajp*), we propose the **MT launching impact rate** to quantify the MT impact on the search ecosystem (e.g. *jp*). MT launching impact rate (I_{MT}) is defined as in Equation 1.

$$I_{MT} = \frac{\Delta S}{\Delta T} \quad (1)$$

$$\Delta S = S_{adapt} - S_{source} \quad (2)$$

$$\Delta T = T_{adapt} - T_{source} \quad (3)$$

where, ΔS is the search result improvement from source queries S_{source} to the query translation from a fine-tuned MT S_{adapt} (as Equation 2), S_{source} and S_{adapt} can be common search metrics such as nDCG, ΔT is the respective translation quality improvement from the source queries T_{source} to the fine-tuned MT query translations T_{adapt} (as equation 3), T_{source} and T_{adapt} can be MT evaluation metrics such as Bleu or Chrf.

We propose the following three groups for language pairs based on their MT impact rate:

- **High-impact language pairs:** Search ecosystems of high-impact language pairs are less tolerant to languages different from the search index language, and more sensitive to the query translation quality. Launching or improving an MT system of those language pairs in the respective search pipeline is more likely to improve the search results.
- **Medium-impact language pairs:** Search ecosystems of medium-impact language pairs are somewhat sensitive to the query translation quality, though not as much as high-impact language pairs.
- **Low-impact language pairs:** Search ecosystems of low-impact language pairs are more robust to different languages and translation quality, and the presence of an MT in the search pipeline has less or little impact on the search result improvement.

3.3 MT improvement impact

We experimented with two improvement strategies for MT in the e-commerce CLIR product search: one is **comprehensive improvement (CI)**, the other is **specific improvement (SI)**. CI usually focuses on the overall improvement in translation quality and targets the entire query traffic. The CI strategies usually involve a change of model architecture or training techniques, etc; SI usually focuses on the improvement of the specific aspects of the query translation quality, and targets a fraction of query traffic. The SI strategies are not necessarily language-agnostic, for example, it can be solving a smaller transliteration problem in a given language, or a brand term preservation improvement for a given language pair.

We propose **The MT improvement impact rate** to quantify the impact of MT comprehensive improvement ($I_{improve}$) on search improvement as in

Equation 4, which can provide signals to choose the right MT improvement strategy for a given language pair.

$$I_{improve} = \frac{\Delta S'}{\Delta T'} \quad (4)$$

$$\Delta S' = S_{adapt} - S_{generic} \quad (5)$$

$$\Delta T' = T_{adapt} - T_{generic} \quad (6)$$

where, $\Delta S'$ is the search result improvement from generic MT query translations $S_{generic}$ to the fine-tuned MT query translations S_{adapt} (as in Equation 5), $S_{generic}$ and S_{adapt} can be the common search metrics such as nDCG; $\Delta T'$ is the respective translation quality improvement from generic MT query translations $T_{generic}$ to the fine-tuned MT query translations T_{adapt} (as in Equation 6), $T_{generic}$ and T_{adapt} can be MT evaluation metrics such as Bleu or Chrf.

Language pairs with higher improvement rate signals both the CI and SI of MT are likely to have positive impact on search. Those with lower rate may benefit more from the focusing on SI for a targeted group of queries from the traffic.

4 Experiment

Language pairs and locales: We selected 12 language pairs from 6 stores for our experiments as seen in Table 1.

Lang pair	Store	Lang pair	Store
esmx-enus	US	ptpt-eses	Spain
ptbr-enus	US	frca-enca	Canada
kokr-enus	US	nlnl-dede	Germany
dede-enus	US	trtr-dede	Germany
mli-enin	India	engb-dede	Germany
knin-enin	India	enus-jajp	Japan

Table 1: Selected 12 language pairs from 6 stores

Test data: The test data is created as described in Section 2. The test set comprises 4000 queries (as reference query translation) per store (e.g. enus), each query is translated into their respective language pairs (e.g. enus -> kokr, enus -> dede). We have also stored the purchased product IDs associated with the queries of the store (e.g. US). We use sampled purchased product ID associated with reference queries as relevant product, and the logarithm of the frequencies of purchased product as the relevance score.

Machine Translation (MT) models: We trained two models per language pair: (i) a *generic MT* system trained on general news and internal crawled

data with (ii) a *domain-specific MT* that is fine-tuned on human translated search queries and synthetically generated query translations through back-translation. These in-house MT models are trained on proprietary data using vanilla transformer architecture (Vaswani et al., 2017) with Sockeye MT toolkit (Domhan et al., 2020).¹

Metric hyper-parameters: We set K to 16 for the top- k search results, using the top-16 products in the search results to compute nDCG@16.

MT metrics: Tables 3 and 4 in the appendix present the MT quality metrics Bleu² and Chrf; Table 5 in the appendix presents search performance metric normalized nDCG@16.³

MT launching and improvement impact rates: With aforementioned metrics, the lower and higher bounds of nDCG@16 of MT impact are presented in Table 6. MT launching impact and improvement rates are computed using nDCG@16 with and Chrf respectively, as in Table 2 in the appendix.

Language pair	MT launching impact		MT improvement impact	
	Δ nDCG/ Δ Bleu	Δ nDCG/ Δ Chrf	Δ nDCG/ Δ Bleu	Δ nDCG/ Δ Chrf
ptpt-eses	0.11	0.15	0.19	0.70
enus-jajp	0.25	0.18	0.78	1.09
engb-dede	0.29	0.32	0.09	0.13
frca-enca	0.31	0.23	0.35	0.60
nlnl-dede	0.47	0.43	0.32	0.69
esmx-enus	0.50	0.34	0.34	0.64
ptbr-enus	0.62	0.56	0.28	1.01
dede-enus	0.62	0.66	0.33	0.61
knin-enin	0.72	0.59	0.19	0.60
trtr-dede	0.85	0.43	0.24	0.43
kokr-enus	0.98	0.49	0.33	0.39
mli-enin	1.04	0.59	0.74	0.72

Table 2: MT launching impact and improvement impact rates

5 Results and Analysis

For the MT launching impact, we rank the language pairs in the descending order according to the MT launching impact rate as well as the impact range respectively, as in Table 7 in the appendix. We observe Bleu and Chrf can give a similar ranking

¹For the purpose of this paper, we are less concerned with the accuracy of the MT models and more interested in the difference in the MT quality as per measured by traditional MT metrics and their evaluation based on our proposed framework. Thus the brevity in the model description.

²SacreBleu version 2.0.0 (Post, 2018)

³Both the nDCG@16 and Chrf are scaled to 0-100 for the computation convenience

with small difference, so the following analysis is based on the MT launching impact from $\Delta nDCG/\Delta\text{Bleu}$ for simplicity. For the MT improvement impact rate, we observe that Bleu makes value scale smaller than Chrf. We will use $\Delta nDCG/\Delta\text{Bleu}$ for the following analysis.

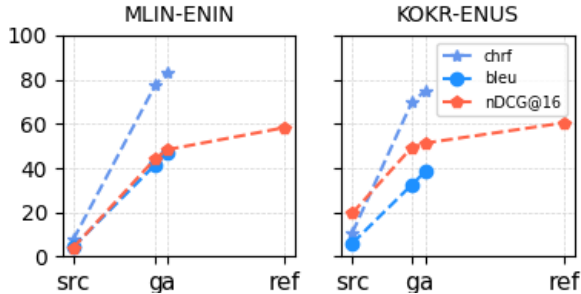


Figure 2: Language pairs where the MT have bigger impact on search pipeline

We observe that MT of language pairs such as *mlin-enin*, *kokr-enus* have higher launching impact rate and should be labeled as the high-impact language pairs. For *mlin-enin*, the MT launching impact rate is 1.04, which signals one point Bleu increase in translation quality can gain slightly more than one point of search improvement. Figure 2 (In Figure 2, 3, 4, “src” refers to source query, “g” refers to generic mt, “a” refers to the adapted (fine-tuned) MT, “ref” refers to the human translation. The axis is scaled according to the Bleu score from 0-100.) illustrates the higher impact language pairs, the range of the MT impact is much bigger, search ecosystems are very responsive to the presence of MT system in the search pipeline, MT and search metrics have similar trending. *mlin-enin* has a much higher improvement rate of 0.74, the ecosystem of the search of this language pair can potentially benefit from both comprehensive improvement (CI) and specific improvement (SI) in the MT. Meanwhile, *kokr-enus* has a much lower improvement rate of 0.33, which signals this search is more likely to benefit from SI than CI.

Language pairs such as *nlnl-dede*, *frca-enca* should be considered as the decent impact language pairs. As illustrated in Figure 3, both have smaller MT impact range and the launching impact rates are high but not quite as the high impact language pairs. As Bleu and Chrf increase from source query to generic MT to fine-tuned MT, nDCG@16 increases slower. Both language pairs have relative lower improvement impact rate which is around 0.3, that signals search of these two language

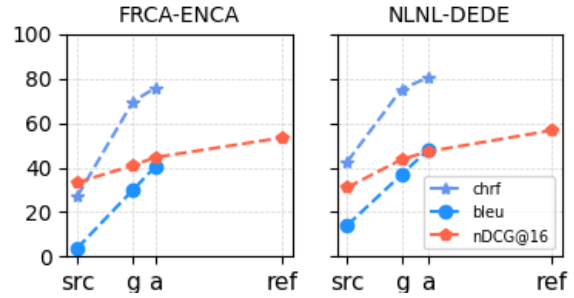


Figure 3: Language pairs where the MT have decent impact on search pipeline

pairs are more likely to benefit from SI than CI. Language pairs such as *ptpt-eses*, *enus-jajp* should

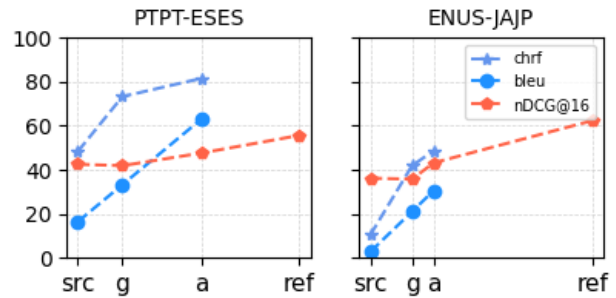


Figure 4: Language pairs where the MT have lower impact on search pipeline

be labeled lower impact language pairs based on the lower launching impact rate. For *ptpt-eses*, one point Bleu increase in translation quality can only achieve 0.11 point of search improvement. Both have smaller launching impact range, thus, search ecosystems are not very responsive to the MT quality improvement. As Bleu and Chrf increase from source query to generic MT to fine-tuned MT, nDCG@16 increases much slower and the trend line is almost flat as Figure 4. In principle, low-impact language arcs might not be prioritized for MT improvement. If there is a need to improve those MT for search, *ptpt-eses* has a much lower MT impact rate of 0.19, so search is likely to benefit from the SI for the MT, whereas *enus-jajp* has much higher improvement rate of 0.78, the search may still benefit from CI as well as SI. Figure 5 and 6 in the appendix are the plots for all other language pairs.

A/B testing: We have also conducted parallel online A/B testing for the following language pairs: *enus-jajp*, *ptpt-eses*, *frca-enca*, *mlin-enin*, *nlnl-dede*, *engb-dede*. For each language pair, we have deployed two fine-tuned MT systems and

integrated them into the search pipeline for the designated store, and the MT system with the comprehensive improvement has higher off-line MT metrics (+5 Bleu points on average) than the baseline model. The A/B testing lasted for 4 weeks on average for all the experiments. For the high impact language pairs, the improved MT systems have seen large increases in business metrics, such as, Order Product Sales (OPS), composite contribution profit (CCP), compared to the baseline model, and have much larger positive impact on the search result quality. For the low impact language pairs, we observe much smaller or even no impact at all. Overall, the A/B testing results are consistent with the MT launching impact rate results we have computed. Moreover, for ptpt-eses and nlnl-dede, we also conducted another round A/B testing with the same experiment setup except using MT with specific improvement to compare with the baseline models. Those two improved MT enhanced the terminology translation of 3-5% of query traffic. The results are consistent with our hypothesis that the MT with SI improvement has much more impact than the MT with CI improvement.

6 Related Work

Machine Translation is necessary to bridge the gap between query translation and cross-lingual information retrieval (Bi et al., 2020). Query translation a key component in large e-commerce stores, previous studies have demonstrated that better translation quality improves retrieval accuracy (Goldfarb et al., 2019; Brynjolfsson et al., 2019).

Queries are naturally short and search engines usually have preferred word choices and collocations based on users' query patterns (Lv and Zhai, 2009; Vechtomova and Wang, 2006). This complicates the evaluation of machine translation for cross-lingual information retrieval in the context of 'fitting in well to the search index'. While machine translation evaluation is well-studied, translation evaluation in downstream task requires more attention especially in the e-commerce cross-lingual information retrieval.

Traditionally, information retrieval evaluation relies on behavioral signals as ground truth to measure relevance of search results; mean reciprocal ranking (MRR), mean average precision (MAP), normalized discounted cumulative gain (nDCG) (Järvelin and Kekäläinen, 2002; Wu et al., 2018;

Nigam et al., 2019b).

Previous studies in cross-lingual information retrieval (CLIR) evaluation relies on pre-annotated datasets that are relatively small and specific to domains outside of e-commerce; for example, the CLEF eHealth test sets (Saleh and Pecina, 2018; Suominen et al., 2018; Zhang et al., 2013) and Wikipedia cross-lingual test set (Sas et al., 2020). Although Sloto et al. (2018) proposed the nDCG-MT metric that leveraged on the reference translation to measure search results relevance, reliance on the ground truth data is still necessary. In pursuit of a more effective approach, we integrate CLIR and MT more closely and evaluate them in an end-to-end task. Our proposed method allows us to fully-automate the evaluation and study the impact of improving MT on CLIR by collecting organic queries in the target language of the e-commerce service and use reference results of these queries as a proxy to human annotation.

7 Conclusion

In this paper, we propose an evaluation framework for MT in the E-commerce multilingual product search through ranking-based search metrics using behavioral signals as proxy relevance information without any human notation, which makes it practical to iteratively improve MT models for the search use case and evaluate them frequently off-line. This framework can also be used to create cross-lingual information retrieval (CLIR) test sets for e-commerce at scale. We also propose a method to measure off-line the MT launching impact and improvement impact rate on search. The former can identify the the high-impact language pairs can be prioritized with more investment in the MT improvement. These experiments can help select the most promising improvement strategy either comprehensive or specific improvement or combination of both to bring a larger impact on the search performance of a given language pair. We have experimented with the proposed evaluation framework and MT impact measuring method on 12 language pairs from 6 stores, and identified the high language pairs of different impact on search and assigned potential improvement strategies. The results are consistent with on-line A/B testing.

References

Tianchi Bi, Liang Yao, Baosong Yang, Haibo Zhang, Weihua Luo, and Boxing Chen. 2020. [Constraint](#)

- translation candidates: A bridge between neural query translation and cross-lingual information retrieval.
- Erik Brynjolfsson, Xiang Hui, and Meng Liu. 2019. Does machine translation affect international trade? evidence from a large digital platform. *Management Science*, 65(12):5449–5460.
- Tobias Domhan, Michael Denkowski, David Vilar, Xing Niu, Felix Hieber, and Kenneth Heafield. 2020. The sockeye 2 neural machine translation toolkit at AMTA 2020. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 110–115, Virtual. Association for Machine Translation in the Americas.
- Avi Goldfarb, Daniel Treffer, et al. 2019. Artificial intelligence and international trade. *The economics of artificial intelligence: an agenda*, pages 463–492.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark. Association for Computational Linguistics.
- Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.
- Zhuolin Jiang, Amro El-Jaroudi, William Hartmann, Damianos Karakos, and Lingjun Zhao. 2020. Cross-lingual information retrieval with BERT. In *Proceedings of the workshop on Cross-Language Search and Summarization of Text and Speech (CLSSTS2020)*, pages 26–31, Marseille, France. European Language Resources Association.
- Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based product retrieval in taobao search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3181–3189.
- Mike Lowndes and Aditya Vasudevan. 2021. Market guide for digital commerce search.
- Hanqing Lu, Youna Hu, Tong Zhao, Tony Wu, Yiwei Song, and Bing Yin. 2021. Graph-based multilingual product retrieval in E-commerce search. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Papers*, pages 146–153, Online. Association for Computational Linguistics.
- Yuanhua Lv and ChengXiang Zhai. 2009. Adaptive relevance feedback in information retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 255–264.
- Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium. Association for Computational Linguistics.
- Jian-Yun Nie. 2010. Cross-language information retrieval. *Synthesis Lectures on Human Language Technologies*, 3(1):1–125.
- Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian (Allen) Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019a. Semantic product search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, KDD '19*, page 2876–2885, New York, NY, USA. Association for Computing Machinery.
- Priyanka Nigam, Yiwei Song, Vijai Mohan, Vihan Lakshman, Weitian, Ding, Ankit Shingavi, Choon Hui Teo, Hao Gu, and Bing Yin. 2019b. Semantic product search.
- 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.
- Andreas Rücklé, Krishnkant Swarnkar, and Iryna Gurevych. 2019. Improved cross-lingual question retrieval for community question answering. In *The World Wide Web Conference, WWW '19*, page 3179–3186, New York, NY, USA. Association for Computing Machinery.
- Shadi Saleh and Pavel Pecina. 2018. Cuni team: Clef ehealth consumer health search task 2018. In *CLEF (Working Notes)*.
- Shadi Saleh and Pavel Pecina. 2020. Document translation vs. query translation for cross-lingual information retrieval in the medical domain. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6849–6860, Online. Association for Computational Linguistics.
- Cezar Sas, Meriem Beloucif, and Anders Søgaard. 2020. WikiBank: Using Wikidata to improve multilingual frame-semantic parsing. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4183–4189, Marseille, France. European Language Resources Association.
- Steve Sloto, Ann Clifton, Greg Hanneman, Patrick Porter, Donna Gates, Almut Silja Hildebrand, and Anish Kumar. 2018. Leveraging data resources for cross-linguistic information retrieval using statistical machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 2: User Track)*, pages 223–233.

Hanna Suominen, Liadh Kelly, Lorraine Goeriot, Aurélie Névéol, Lionel Ramadier, Aude Robert, Evangelos Kanoulas, Rene Spijker, Leif Azzopardi, Dan Li, et al. 2018. Overview of the clef ehealth evaluation lab 2018. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 286–301. Springer.

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.

Olga Vechtomova and Ying Wang. 2006. A study of the effect of term proximity on query expansion. *Journal of Information Science*, 32(4):324–333.

Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. [Turning clicks into purchases: Revenue optimization for product search in e-commerce](#). SIGIR '18, page 365–374, New York, NY, USA. Association for Computing Machinery.

Bryan Zhang. 2022. [Improve MT for search with selected translation memory using search signals](#). In *Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track)*, pages 123–131, Orlando, USA. Association for Machine Translation in the Americas.

Hang Zhang and Liling Tan. 2021. [Textual representations for crosslingual information retrieval](#). In *Proceedings of The 4th Workshop on e-Commerce and NLP*, pages 116–122, Online. Association for Computational Linguistics.

Lei Zhang, Achim Rettinger, Michael Färber, and Marko Tadić. 2013. A comparative evaluation of cross-lingual text annotation techniques. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 124–135. Springer.

A Appendix

sacreBleu			
Language pair	source	generic MT	adapted MT
trtr-dede	6.4	23.4	28.8
enus-jajp	2.8	21.1	30.6
esmx-enus	2.6	26.6	33.3
kokr-enus	6.02	32.53	38.39
frca-enca	3.77	30.01	40.46
ptbr-enus	3.7	26.8	41.91
mlln-enin	4.41	41.7	47.02
nl-nl-dede	14.09	36.87	48.11
dede-enus	6.88	46.74	60.93
ptpt-eses	16.49	33.28	63.08
engb-dede	10.1	45.61	63.08
knin-enin	2.77	52.02	71.27

Table 3: MT metric - Bleu for source queries and query MT translations

Chrf			
language pair	source	generic mt	adapted mt
dede-enus	30.49	73.82	81.36
engb-dede	33.08	69.68	80.99
enus-jajp	10.49	41.91	48.67
esmx-enus	24.92	65.62	69.19
frca-enca	27.04	69.72	75.85
kokr-enus	10.7	69.79	74.75
mlln-enin	7.64	77.7	83.19
nl-nl-dede	42.63	75.34	80.58
ptbr-enus	25.66	64.2	68.33
ptpt-eses	48.37	73.26	81.52
trtr-dede	23.08	64.27	67.3
knin-enin	5.29	82.67	88.62

Table 4: MT metric -Chrf for source queries and query MT translations

nDCG@16				
Language pair	source	generic MT	adapted MT	ref
enus-jajp	36.2	35.80	43.19	62.30
frca-enca	33.34	40.98	44.64	53.47
trtr-dede	26.8	44.60	45.90	63.90
nlnd-dede	31.11	43.67	47.26	56.76
ptpt-eses	42.53	41.89	47.64	55.65
mlln-enin	4.00	44.38	48.34	58.28
ptbr-enus	27.2	46.71	50.89	60.28
kokr-enus	19.59	49.38	51.29	60.42
dede-enus	17.78	46.91	51.54	60.27
knin-enin	2.90	48.7	52.27	58.28
esmx-enus	37.7	50.6	52.90	69.40
engb-dede	38.54	52.38	53.88	61.91

Table 5: search metric (nDCG@16) of source queries and query MT and reference translations

Language pair	lower bound	upper bound	impact range
ptpt-eses	42.53	55.65	13.12
frca-enca	33.34	53.47	20.13
engb-dede	38.54	61.91	23.37
nlnd-dede	31.11	56.76	25.65
enus-jajp	36.20	62.30	26.10
esmx-enus	37.70	69.40	31.70
ptbr-enus	27.20	60.28	33.08
trtr-dede	26.80	63.90	37.10
kokr-enus	19.59	60.42	40.83
dede-enus	17.78	60.27	42.49
mlln-enin	4.00	58.28	54.28
knin-enin	2.90	58.28	55.38

Table 6: The MT impact range (nDCG@16)

Rank	impact range	MT launching impact	
		Δ nDCG/ Δ Bleu	Δ nDCG/ Δ Chrf
1	knin-enin	mlln-enin	dede-enus
2	mlln-enin	kokr-enus	knin-enin
3	dede-enus	trtr-dede	mlln-enin
4	kokr-enus	knin-enin	ptbr-enus
5	trtr-dede	dede-enus	kokr-enus
6	ptbr-enus	ptbr-enus	trtr-dede
7	esmx-enus	esmx-enus	nlnd-dede
8	enus-jajp	nlnd-dede	esmx-enus
9	nlnd-dede	frca-enca	engb-dede
10	engb-dede	engb-dede	frca-enca
11	frca-enca	enus-jajp	enus-jajp
12	ptpt-eses	ptpt-eses	ptpt-eses

Table 7: Language pair ranking based on the MT launching impact

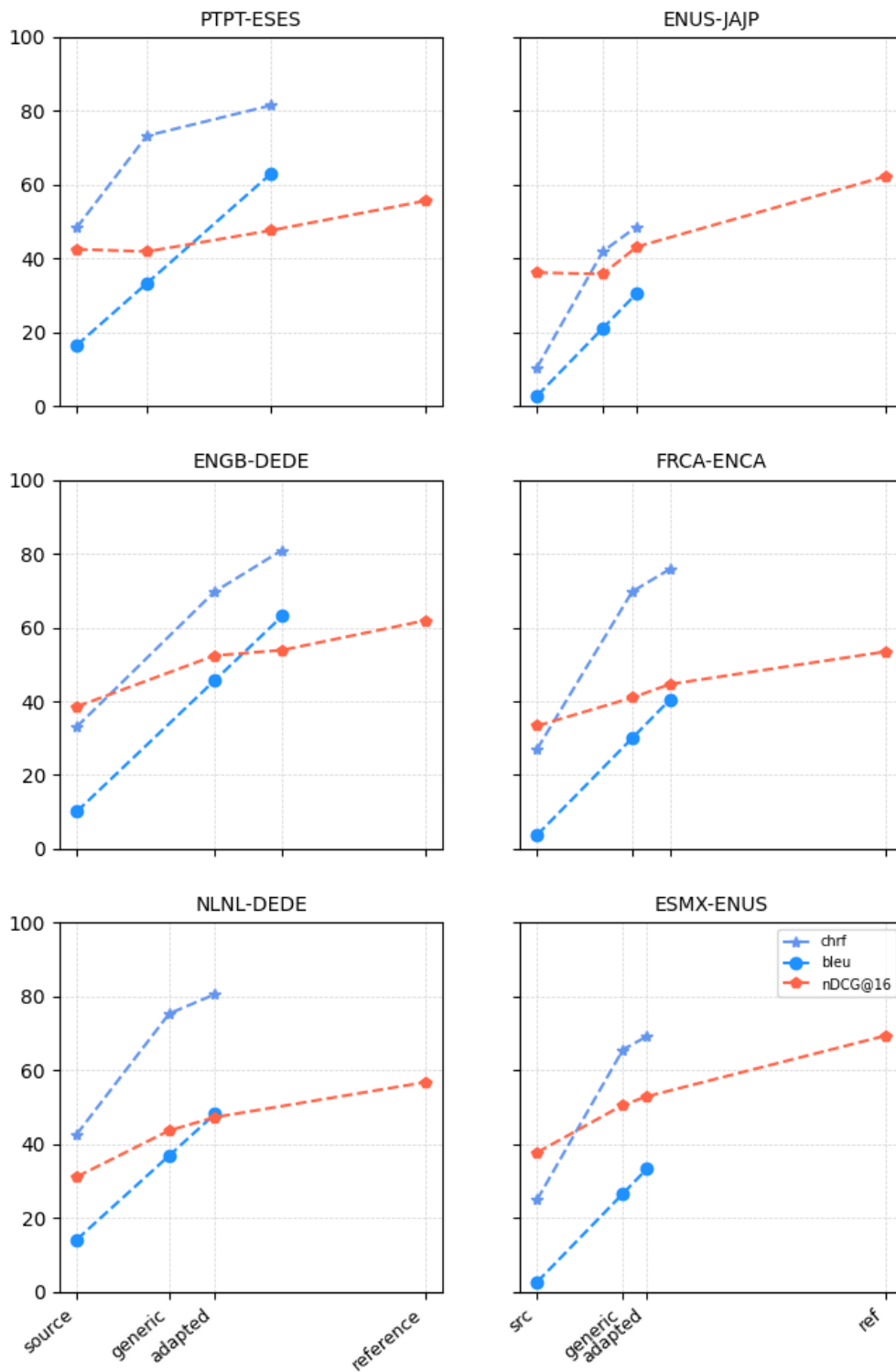


Figure 5: MT quality metrics and search metrics.png

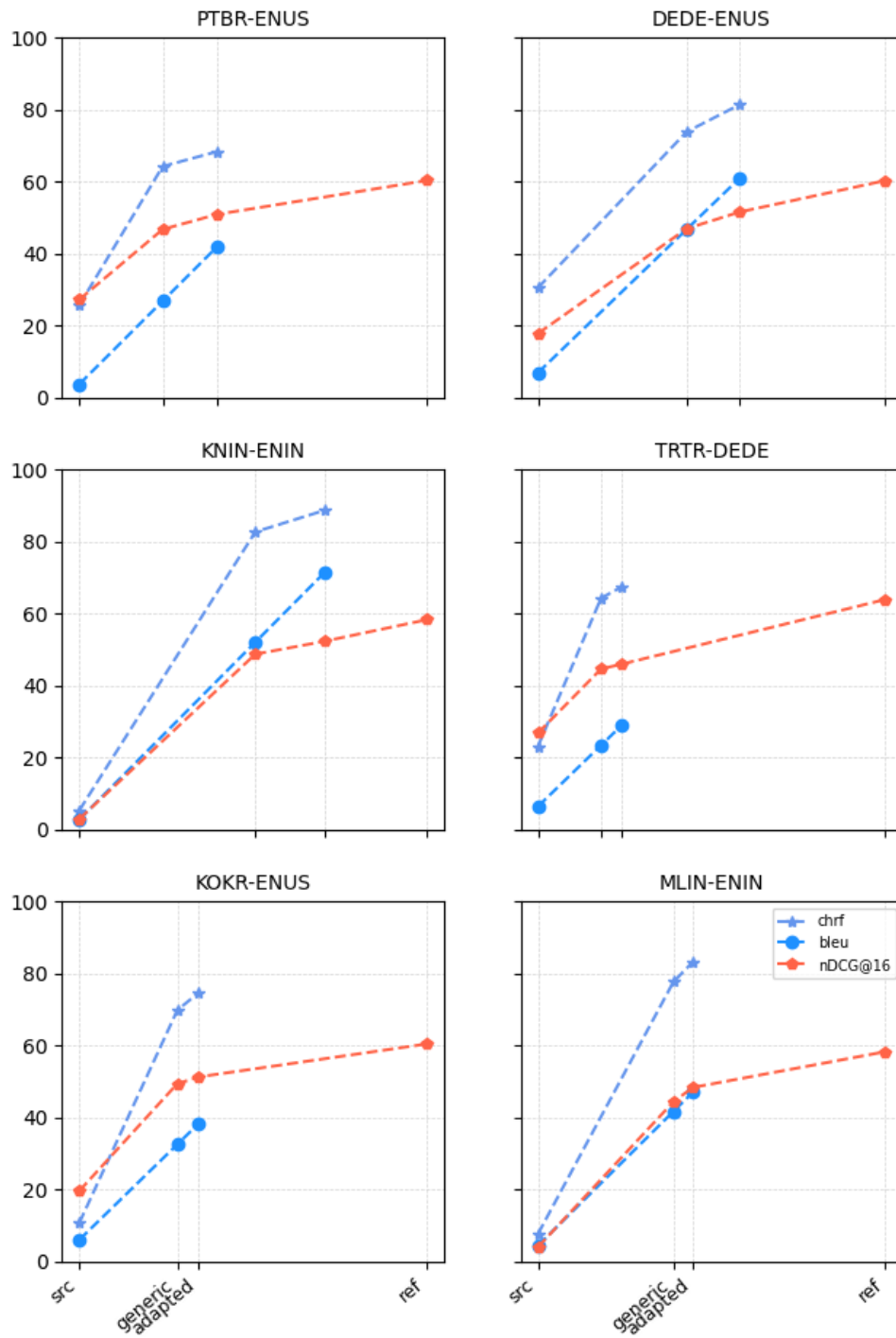


Figure 6: MT quality metrics and search metrics

Ask-and-Verify: Span Candidate Generation and Verification for Attribute Value Extraction

Yifan Ding^{1*}, Yan Liang², Nasser Zalmout², Xian Li², Christian Grant³, Tim Weninger¹

University of Notre Dame¹, Amazon.com², University of Oklahoma³,

{yding4, tweninge}@nd.edu, {ynliang, nzalmout, xianlee}@amazon.com, cgrant@ou.edu

Abstract

The product attribute value extraction (AVE) task aims to capture key factual information from product profiles, and is useful for several downstream applications in e-Commerce platforms. Previous contributions usually formulate this task using sequence labeling or reading comprehension architectures. However, sequence labeling models tend to be conservative in their predictions resulting in a high false negative rate. Existing reading comprehension formulations, on the other hand, can over-generate attribute values which hinders precision. In the present work we address these limitations with a new end-to-end pipeline framework called Ask-and-Verify. Given a product and an attribute query, the Ask step detects the top-K span candidates (*i.e.*, possible attribute values) from the product profiles, then the Verify step filters out false positive candidates. We evaluate Ask-and-Verify model on Amazon’s product pages and AliExpress public dataset, and present a comparative analysis as well as a detailed ablation study. Despite its simplicity, we show that Ask-and-Verify outperforms recent state-of-the-art models by up to 3.1% F1 absolute improvement points, while also scaling to thousands of attributes.

1 Introduction

The product profiles in e-Commerce platforms are usually comprised of free-form natural language description of the main product features. The product attribute value extraction (AVE) task is used to extract key factual information from textual product descriptions. Properly extracted attribute values can facilitate several downstream applications, such as search (Xiao et al., 2021), recommendation systems (Hwangbo et al., 2018), and task-oriented dialogue systems (Yan et al., 2017). In the various retail categories there are millions of different

* Most of the work was done during an internship at Amazon.



Figure 1: An example of attribute value extraction task on a dairy product. Corresponding attribute values are extracted for several different attributes including flavor, target age, brand, gluten, and organic information.

product types with thousands of unique attributes, so AVE should ideally be scalable with respect to the number of attributes, providing high coverage for all possible values, while maintaining accurate overall predictions. Fig. 1 shows an example for the AVE task on a dairy product. In this case, AVE aims to extract the corresponding attribute values of multiple product attributes including *flavor*, *target age*, *gluten information*, among others.

AVE is a central organizational task in online shopping systems, significant attention has been paid to the task resulting in a handful of highly-optimized systems (Zalmout et al., 2021; Yan et al., 2021; Lin et al., 2021; Wang et al., 2020; Xu et al., 2019; Zheng et al., 2018). Most of these models use either a sequence labeling formulation, or a machine reading comprehension (MRC) formulation. Sequence labeling is a popular formulation in the named entity recognition literature. However, its application on the AVE task tends to generate conservative outputs, resulting in many false negatives. This is mostly caused by an overabundance of negative token labels (*i.e.*, the ‘O’ in BIOES schema).

Recently, the AVEQA model (Wang et al., 2020) addressed the AVE task using a reading compre-

hension formulation (*i.e.*, question answering – hence AVEQA). This formulation tends to be more flexible and scalable than sequential labelling approaches, however, we observed that AVEQA tends to over-generate irrelevant outputs and does not generalize to multiple attribute values.

In the present work, we address these limitations in existing systems with a new end-to-end framework we dub Ask-and-Verify, consisting of a span candidate generation step (Ask) and a span verification step (Verify). The Ask step first identifies relevant span candidates by locating potential boundaries (*i.e.* starting and ending indices) with two individual multi-label classifiers based on token features. The goal of the Verify step is to eliminate irrelevant span candidates with span features. The overall framework is attribute-agnostic, which can capture salient attribute information from the input sequence, and can generalize to thousands of attributes without attribute-specific parameters.

In summary, we present the Ask-and-Verify framework, which disentangles the attribute value extraction task into an end-to-end pipeline of (1) span candidate generation and (2) verification. We design the multi-label classifiers and span candidate collection module to obtain valid high-quality span candidates within the model. The verification module is an attribute-agnostic binary classifier based on span features. Through extensive experiments on two real-world E-commerce datasets, we show that the Ask-and-Verify framework outperforms the current crop of state-of-the-art models, and is able to scale to thousands of attributes.

2 Related Work

2.1 Attribute Value Extraction

The goal of the attribute value extraction task is to extract key factual information about a product from its text description. Recent contributions typically formulate the AVE task as a sequence labeling task (Yan et al., 2021; Karamanolakis et al., 2020; Xu et al., 2019; Zheng et al., 2018). The main idea is to assign token-wise attribute labels with context-aware token features. To extract different attributes, multiple strategies have been presented in previous works. OpenTag (Zheng et al., 2018) utilized separate tag-sets for each attribute, SuOpenTag (Xu et al., 2019) and Adatag (Yan et al., 2021) utilized single tag-set for all the attributes while attribute information is explicitly injected at the encoder or decoder. Recently, AVEQA (Wang et al., 2020)

utilizes machine reading comprehension to extract attribute values by treating attributes as questions and text descriptions as the passage.

2.2 Machine Reading Comprehension

Machine reading comprehension (MRC) is a general task within the fields of information retrieval (IR) and natural language processing (NLP), which aims to find correct answers to a question in a given passage (Rajpurkar et al., 2016, 2018; Zhang et al., 2020). Illustrated via questions at the bottom of Fig. 1, the AVE task can naturally be formulated as an MRC task, which is to extract correct attribute values (answer) within a product text description (passage) for a given attribute query (question). One complication is that the AVE task must handle unanswerable attribute queries (unanswerable questions) and multiple attribute values for an attribute (multiple answers to a single question).

2.3 Candidate Generation and Selection

Candidate generation and selection is widely used in object detection (Carion et al., 2020; Ren et al., 2015) and instance segmentation (Wang et al., 2021; He et al., 2017) in computer vision. The candidate generation step generates candidate bounding boxes which can carry instance information used in the selection step. Recently, NLP researchers have developed span-based models (Shen et al., 2021; Joshi et al., 2020; Yamada et al., 2020; Li et al., 2020) to obtain state-of-the-art performance on span-based or entity-centered tasks like named entity recognition (Ding et al., 2021; Huang et al., 2015), entity linking (Ding et al., 2022; Botzer et al., 2022), and machine reading comprehension (Rajpurkar et al., 2018) among others. A key insight of the Ask-and-Verify framework is to show that this kind-of candidate generation and selection formulation used widely in computer vision can also be used to benefit the the AVE task and potentially other span-level NLP tasks.

3 Methodology

Task Definition: Given a product description $X = [x_1, x_2, \dots, x_L]$ with L tokens, and an attribute A from a pre-defined attribute set \mathcal{A} , the AVE task aims to extract all of the unique attribute values $\mathcal{Y} = \{y_1, y_2, \dots, y_M\}$ corresponding to A . Each attribute value y_m is composed of one or more consecutive tokens within X . If no proper attribute values is found in X for A , an empty set should

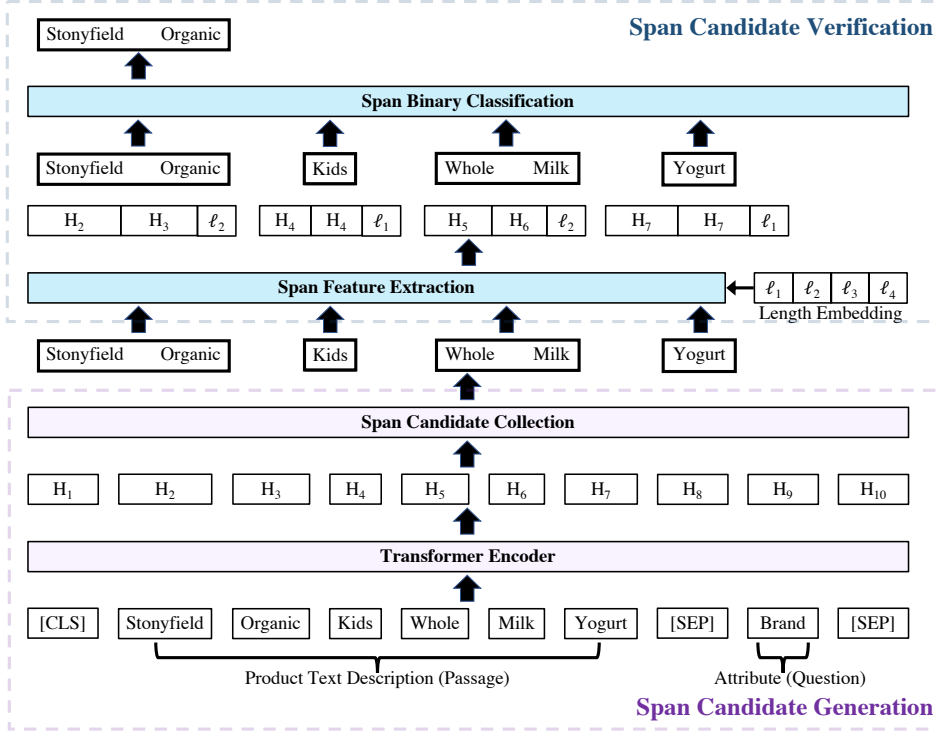


Figure 2: Overview of the Ask-and-Verify framework, a two-step attribute value extraction framework with *Span Candidate Generation* (Ask) and *Span Candidate Verification* (Verify). The framework takes input sequences of entity text descriptions and a single attribute of interest. In the *Span Candidate Generation* step, the input sequence is first passed to a Transformer Encoder to obtain hidden states. The Span Candidate Collection module processes the hidden states to obtain top-K valid span candidates. In the *Span Candidate Verification* step, span embeddings composed of start-token hidden states, end-token hidden states and a span-length embedding, are obtained in Span Feature Extraction module for each generated span candidate. Finally, the span embedding is passed to the Span Binary Classification module to obtain the extracted attribute values.

be returned for \mathcal{Y} . Following common practice in question answering, we also call this case as unanswerable case.

Ask-and-Verify: Our framework addresses the AVE task in an end-to-end manner with two major components: (1) span candidate generation (Ask), and (2) span candidate verification (Verify). For an attribute of interest A , the first step generates the potential span candidates. The second verification step filters the candidates and selects a subset.

3.1 Span Candidate Generation

This step generates potential span candidates. Specifically, we employ two individual multi-label classifiers to locate starting index and ending index of span candidates. Formally, given a product with text description X and an attribute of interest A , we use a sub-word tokenizer to tokenize original tokens along with the attribute into sub words SW :

$$SW = \text{Tokenizer}(\{[\text{CLS}], X, [\text{SEP}], A\}) \quad (1)$$

Following common practice, we pad the sequences to some fixed length K and longer se-

quences are also fixed to the same length K by truncating the tokens of text description. The processed tokens are then fed into a **BERT** encoder to obtain d -dimensional hidden states $H \in \mathcal{R}^{K \times d}$:

$$H = \text{Encoder}(SW) \quad (2)$$

The hidden states H_s of index s are further fed through a linear layer and Softmax to obtain the probabilities for the starting token. Similarly, the probability index e for the ending token is obtained by feeding the corresponding hidden states H_e through another linear layer and Softmax.

$$P_{\text{start}}^\theta(s|X, A) = \frac{\exp(w_{\text{start}}^T H_s)}{\sum_{k=1}^K \exp(w_{\text{start}}^T H_k)} \quad (3)$$

$$P_{\text{end}}^\theta(e|X, A) = \frac{\exp(w_{\text{end}}^T H_e)}{\sum_{k=1}^K \exp(w_{\text{end}}^T H_k)} \quad (4)$$

At the training stage, two K -class (K is the fixed length of input sentence) classifiers are used individually on start indexes and end indexes with multi-label cross-entropy loss (see Eq. (5)-(6)).

Note that start token(s) of \mathcal{Y} forms a set \mathcal{Y}_S , and end token(s) of \mathcal{Y} forms a set \mathcal{Y}_E . Any correct token in \mathcal{Y}_S is considered as a positive starting token and any correct token in \mathcal{Y}_E is considered as a positive ending token.

$$\mathcal{L}_{\text{start}} = - \sum_{s=1}^K \mathbb{1}(SW_s \in \mathcal{Y}_S) \log P_{\text{start}}^\theta(s|X, A) \quad (5)$$

$$\mathcal{L}_{\text{end}} = - \sum_{e=1}^K \mathbb{1}(SW_e \in \mathcal{Y}_E) \log P_{\text{end}}^\theta(e|X, A) \quad (6)$$

To obtain the actual span candidates \mathcal{M} , each span candidate $m_{(s,e)}$ has to have top-K probabilities within valid spans (see Eq. (7)). A span is a valid if and only if all the span tokens are within the range of text description with positive lengths up to T tokens (see Eq. (8)). Note that span candidate generation is part of the model thus span candidates are obtained in both training and inference stage.

$$\mathcal{M} = \{m_{(s,e)} \mid \mathbf{Mask}_{(s,e)} \wedge P_{\text{start}}^\theta(s|X, A) + P_{\text{end}}^\theta(e|X, A) \in \mathbf{top-K}\} \quad (7)$$

$$\mathbf{Mask}_{(s,e)} = \mathbb{1}(SW_s \in X \wedge SW_e \in X \wedge 1 \leq e - s \leq T) \quad (8)$$

3.2 Span Candidate Verification

The span candidate verification step aims to verify each span candidate generated from previous step, and choose the final attribute value extraction output. We utilize a simple but effective uniform binary classification model for the verification step. We make individual binary (*i.e.*, yes/no) classifications for each span candidate with the corresponding span-level features.

Formally, given the same hidden state H from Eq. (2), each span candidate $m_{(s,e)}$ obtains its span features $H_{m_{(s,e)}}$ by concatenating starting token’s hidden state H_s , ending token’s hidden state H_e , and span candidate’s length embedding ℓ_{e-s} .

$$H_{m_{(s,e)}} = \text{Concat}([H_s; H_e; \ell_{e-s}]) \quad (9)$$

$$\hat{H}_{m_{(s,e)}} = \mathbf{DropOut}(\mathbf{ReLU}(w_1^T H_{m_{(s,e)}})) \quad (10)$$

$$P_{\text{span}}^\theta(m_{(s,e)}|X, A) = \frac{\exp(w_2^T \hat{H}_{m_{(s,e)}})}{\sum_{m' \in \mathcal{M}} \exp(w_2^T \hat{H}_{m'})} \quad (11)$$

where $\ell \in \mathcal{R}^{T \times d_F}$ is the learned span length features with d_F dimension. The span features are then fed into a single-layer feed forward neural network with DropOut and a ReLU layer to obtain the corresponding span state $\hat{H}_{m_{(s,e)}}$. $\hat{H}_{m_{(s,e)}}$ further goes through another linear layer and Softmax to obtain the probabilities.

The objective function $\mathcal{L}_{\text{span}}$ of the verification step is the sum of the binary cross entropy losses for each span candidate $m_{(s,e)}$. A span is positive if and only if it exactly matches one of the ground truth attribute value(s).

$$\mathcal{L}_{\text{span}} = - \sum_{m \in \mathcal{M}} \left(\mathbb{1}(m \in \mathcal{Y}) \log P(m|X, A) + \mathbb{1}(m \notin \mathcal{Y}) \log(1 - P(m|X, A)) \right) \quad (12)$$

$$\mathcal{L} = \mathcal{L}_{\text{start}} + \mathcal{L}_{\text{end}} + \mathcal{L}_{\text{span}} \quad (13)$$

3.3 Training and Inference

The training objective function of the Ask-and-Verify framework is the sum of the starting index loss, ending index loss, and the span binary classification loss (see Eq. (13)). During inference, each span candidate \hat{m} is ranked according to its binary classification score $P_{\text{span}}^\theta(\hat{m}|X, A)$. The span candidate with higher score than some threshold value τ makes it to the ranking step. In the ranking order, a span candidate is selected if it does not have any overlapping token(s) with any of the already selected spans. If no spans make it to the ranking step, then an empty set is returned. Additional details (*i.e.* hyperparameters) can be found in the reproducibility section A of the appendix.

4 Experiments

We conduct extensive experiments using the AliExpress (Xu et al., 2019) and Amazon datasets and compare the Ask-and-Verify framework with ten state-of-the-art methods.

4.1 Datasets

AliExpress: We use the public version of the AliExpress dataset (**AE-110K**). Following previous work (Wang et al., 2020), we randomly partition the product-instances into an 80/20 train/test split for scaling experiments. Additionally, we also focus on the 50 most frequent attributes, but remove 2

Table 1: Test macro precision (P), recall (R) and F1 scores on AE-48, AZ-15 and AZ-33. Best scores are highlighted in bold, second best scores are underlined.

Model	AE-48			AZ-15			AZ-33		
	P	R	F ₁	P	R	F ₁	P	R	F ₁
BiLSTM (Huang et al., 2015)	0.788	0.771	0.776	0.742	0.519	0.593	0.731	0.511	0.586
BERT (Devlin et al., 2019)	0.787	0.814	0.800	0.720	0.506	0.582	0.736	0.509	0.589
OpenTag (Zheng et al., 2018)	-	-	-	0.751	0.519	<u>0.594</u>	0.708	0.482	0.557
SuOpenTag (Xu et al., 2019)	<u>0.806</u>	0.795	0.798	0.749	0.503	0.585	0.711	0.533	0.593
AdaTag (Yan et al., 2021)	0.801	0.805	0.799	0.751	0.518	0.591	0.712	<u>0.542</u>	<u>0.599</u>
AVEQA (Wang et al., 2020)	<u>0.806</u>	0.807	<u>0.804</u>	0.618	0.512	0.551	0.633	0.523	0.563
MRC-For-NER (Li et al., 2020)	0.753	0.800	0.774	0.562	0.428	0.470	0.652	0.482	0.543
W2NER (Li et al., 2022)	-	-	-	0.847	0.304	0.405	0.838	0.271	0.369
Locate-and-Label (Shen et al., 2021)	0.713	0.673	0.669	0.655	0.564	0.569	0.697	0.512	0.549
Sequence-to-Set (Tan et al., 2021)	0.778	0.621	0.665	<u>0.786</u>	0.411	0.501	<u>0.763</u>	0.420	0.501
Ask-and-Verify	0.821	<u>0.813</u>	0.814	0.750	<u>0.551</u>	0.625	0.744	0.562	0.629

that fail on the AdaTag model (Yan et al., 2021). This setting is referred to as **AE-48** dataset.

Amazon: Similar to previous work (Yan et al., 2021), we collected datasets from Amazon’s product pages. The raw training data includes 33 frequent attributes and 745,216 total samples. Example attributes including color, flavor, skin type, hair type, pattern type, and age range description. Test data is annotated by Amazon employees, including 15 attributes (from the 33 attributes) and 11,000 total samples. We consider two experiment settings: first we use all 33 attributes (**AZ-33**) from the training set; in the second setting we restrict the training instances to include at least one of 15 attributes present in test set (**AZ-15**).

4.2 Existing Models

We compared the AVE task performance of the Ask-and-Verify against ten state-of-the-art models including two standard sequential labeling models: BiLSTM (Huang et al., 2015) and BERT (Devlin et al., 2019); four state-of-the-art attribute value extraction models: OpenTag (Zheng et al., 2018), SuOpenTag (Xu et al., 2019), AdaTag (Yan et al., 2021), and AVEQA (Wang et al., 2020); four state-of-the-art named entity recognition models: MRC-for-NER (Li et al., 2020), W2NER (Li et al., 2022), Locate-and-Label (Shen et al., 2021), and Sequence-to-Set (Tan et al., 2021).

4.3 Metrics

Following the previous contributions, we use the exact entity matching criteria for evaluation. A predicted attribute value is considered to be a true

Table 2: Test micro precision (P), recall (R) and F1 scores on AE-110k. Many models are not included because they could not scale to the large size of attributes in the AE-110K dataset. Best scores are highlighted in bold. ♣: our reported scores are different from original reported values, details can be found in section B and D of appendix.

Model	AE-110K		
	P	R	F ₁
SuOpenTag (Xu et al., 2019)	0.641	0.575	0.607
AVEQA (Wang et al., 2020) ♣	0.784	0.711	0.746
Ask-and-Verify	0.798	0.723	0.759

positive if and only if it exactly matches one of the ground truth values. In the main experiments shown in Table 1 we compute macro precision (P), recall (R) and F1 scores (F1) by aggregating across testing attributes. In the scaling experiment shown in Table 2, we use micro precision, recall and F1 scores following previous work (Wang et al., 2020).

4.4 Results

The results of our principal experiments over the three E-Commerce dataset settings (AE-48, AZ-15 and AZ-33) are listed in Table 1. We observe that the Ask-and-Verify framework outperforms the existing methods on all three settings with absolute improvements in the F1 score of +1.0%, +3.1% and +3.0% respectively, compared to second best baseline. Specifically, Ask-and-Verify obtains the best or second best recall scores. As for precision, only state-of-the-art NER models obtain higher precision than Ask-and-Verify but with the cost of much

Table 3: Ablation study of different model choices on span candidate generation (Ask) and verification (Verify). Best scores are highlighted in bold.

	AZ-15			AZ-33		
	P	R	F ₁	P	R	F ₁
Ask-and-Verify	0.750	0.551	0.625	0.744	0.562	0.629
w/o Verify	0.279	0.601	0.374	0.328	0.618	0.421
Verify w/ PURE	0.825	0.316	0.418	0.829	0.287	0.388
Ask w/ n-gram	0.717	0.510	0.587	0.708	0.507	0.582
Ask w/ nouns	0.611	0.264	0.361	0.580	0.266	0.357

more downgrade on the recall.

We further conduct experiments to test the scalability of the Ask-and-Verify framework on the AE-110K setting with more than two thousand attributes. Results of this experiment are listed in Table 2. We compared Ask-and-Verify with AVEQA and SuOpenTag only because the other models are not able to scale up to all attributes. We find the AVEQA model shows significantly better scalability compared to the SuOpenTag. Compared to AVEQA, the Ask-and-Verify framework is able to further boost the performance, with improvements in precision, recall and F1 scores at +1.4%, +1.2% and 1.3% respectively.

5 Ablation Studies

5.1 Effectiveness of the Ask-Step

We study the effect of using alternative formulations for the Ask step to provide span candidates. Intuitively, we can remove the Ask component and replace it with a basic n-grams setup. Specifically, we use n-grams and noun phrase chunking as alternatives for the Ask step. For the n-grams model, we consider all the possible spans up to five words. For the noun-phrase chunking model, we utilize spaCy’s chunker¹ to extract all the nouns of the input text. We keep the same Verification step and conduct experiments on AZ-15 and AZ-33 settings. As shown in Table 3, both n-grams and noun phrase chunking show a lower performance compared to the Ask component. Using n-grams drops f1 by 3.8% on AZ-15, and 4.7% on AZ-33. While noun phrases result in a larger drop, by more than 20% on both settings.

5.2 Effectiveness of the Verification-Step

To study the effect of the Verification step, we first consider removing the Verification step completely,

¹<https://github.com/explosion/spaCy>

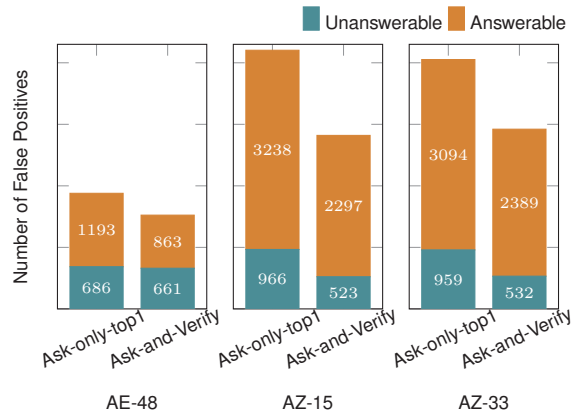


Figure 3: Number of false positive samples in Ask-only and Ask-and-Verify settings. The Verify-step substantially reduces false positives.

and use the span candidates directly as the output. As shown in Table 3, removing the verification step leads to significant precision drop of about 47.1% and 41.6%, with moderate recall improvement of 5.0% and 5.6% on the AZ-15 and AZ-33. We also replace the Verification step with the PURE (Zhong and Chen, 2021) model. Even though the alternative can improve precision with 7.5% and 8.5%, it drops recall by about 23.5% and 27.5%.

Further, to quantitatively understand how the Verify model can reduce the number of false positives presented in the span candidates, we count the false positives in the Ask-and-Verify output and Ask-only-top1 span output. Compared to Ask-only-top-1, the addition of the Verify model in our framework can significantly reduce the false positives in both answerable and unanswerable cases. In the AE-48 setting, answerable false positive samples are reduced by about 20%. On the AZ-15 and AZ-33 settings, Ask-and-Verify can filter-out more than 40% of the unanswerable cases, and more than 20% of the answerable cases.

6 Deployment Considerations

The Ask-and-Verify framework is currently under deployment evaluation. Investigation is going on to verify if the framework can be deployed with minimal changes of the existing workflow. In our deployment tests, we found that the Ask-and-Verify framework has better precision and recall performance. Ask-and-Verify is also flexible, and can adapt to a wide application scenarios that might require varying precision and recall levels, by changing the threshold values. Moreover, a single model can cover a large number of attributes – this is par-

ticularly important since E-commerce platforms can hold billions of different products with thousands of attributes.

7 Conclusions

In this paper we described a new end-to-end framework, Ask-and-Verify, for the attribute value extraction task. This framework has two main components: (1) a span candidate generation step, and (2) a verification step. The span candidate generation step can provide high-quality span candidates and the verification step can further remove irrelevant span candidates. Ask-and-Verify utilizes two individual multi-label classifiers in the candidate span generation step and an attribute agnostic span-based binary classifier in the verification step. We performed a comparative analysis on an Amazon products dataset as well as a publicly available dataset from AliExpress. We evaluate Ask-and-Verify compared to ten other baseline models. Despite its simplicity, Ask-and-Verify consistently outperforms these state-of-the-art methods, and is able to scale up to thousands of unique attributes. Ask-and-Verify also has high flexibility and allows for effective threshold tuning.

Acknowledgements

This project was funded in part by DARPA under contract HR001121C0168 and HR00112290106. We would like to thank Chenwei Zhang and Jun Ma from Amazon, for their constructive feedback on the work. We would also like to thank the anonymous reviewers for their valuable comments.

References

- Nicholas Botzer, Yifan Ding, and Tim Weninger. 2022. Reddit entity linking dataset. *Information Processing and Management*, 58(3).
- Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. 2020. End-to-end object detection with transformers. In *2020 European Conference on Computer Vision*, pages 213–229, Cham. Springer International Publishing.
- 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.
- Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.
- Yifan Ding, Nicholas Botzer, and Tim Weninger. 2022. Posthoc verification and the fallibility of the ground truth. In *Proceedings of the First Workshop on Dynamic Adversarial Data Collection*, pages 23–29, Seattle, WA. Association for Computational Linguistics.
- Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv:1508.01991*.
- Hyunwoo Hwangbo, Yang Sok Kim, and Kyung Jin Cha. 2018. Recommendation system development for fashion retail e-commerce. *Electronic Commerce Research and Applications*, 28:94–101.
- 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.
- Giannis Karamanolakis, Jun Ma, and Xin Luna Dong. 2020. TXtract: Taxonomy-aware knowledge extraction for thousands of product categories. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8489–8502, Online. Association for Computational Linguistics.
- Jingye Li, Hao Fei, Jiang Liu, Shengqiong Wu, Meishan Zhang, Chong Teng, Donghong Ji, and Fei Li. 2022. Unified named entity recognition as word-word relation classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10965–10973.
- 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, Online. Association for Computational Linguistics.
- Rongmei Lin, Xiang He, Jie Feng, Nasser Zalmout, Yan Liang, Li Xiong, and Xin Luna Dong. 2021. Pam: Understanding product images in cross product category attribute extraction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery*

- and Data Mining, page 3262–3270, New York, NY, USA. Association for Computing Machinery.
- 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.
- 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.
- Yongliang Shen, Xinyin Ma, Zeqi Tan, Shuai Zhang, Wen Wang, and Weiming Lu. 2021. Locate and label: A two-stage identifier for nested named entity recognition. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2782–2794, Online. Association for Computational Linguistics.
- Zeqi Tan, Yongliang Shen, Shuai Zhang, Weiming Lu, and Yueting Zhuang. 2021. A sequence-to-set network for nested named entity recognition. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 3936–3942. International Joint Conferences on Artificial Intelligence Organization. Main Track.
- Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. Learning to extract attribute value from product via question answering: A multi-task approach. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 47–55, New York, NY, USA. Association for Computing Machinery.
- Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. 2021. End-to-end video instance segmentation with transformers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8737–8746.
- Liqiang Xiao, Jun Ma, Xin Luna Dong, Pascual Martínez-Gómez, Nasser Zalmout, Wei Chen, Tong Zhao, Hao He, and Yaohui Jin. 2021. End-to-end conversational search for online shopping with utterance transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3477–3486, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Huimin Xu, Wenting Wang, Xin Mao, Xinyu Jiang, and Man Lan. 2019. Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5214–5223, Florence, Italy. Association for Computational Linguistics.
- Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. Association for Computational Linguistics.
- Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. AdaTag: Multi-attribute value extraction from product profiles with adaptive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4694–4705, Online. Association for Computational Linguistics.
- Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. 2017. Building task-oriented dialogue systems for online shopping. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.
- Nasser Zalmout, Chenwei Zhang, Xian Li, Yan Liang, and Xin Luna Dong. 2021. All you need to know to build a product knowledge graph. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD ’21, page 4090–4091, New York, NY, USA. Association for Computing Machinery.
- Zhuosheng Zhang, Hai Zhao, and Rui Wang. 2020. Machine reading comprehension: The role of contextualized language models and beyond. *arXiv:2005.06249*.
- Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1049–1058, New York, NY, USA. Association for Computing Machinery.
- Zexuan Zhong and Danqi Chen. 2021. A frustratingly easy approach for entity and relation extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 50–61, Online. Association for Computational Linguistics.

A Reproducibility

Ask-and-Verify is implemented using the PyTorch and transformers packages based on the uncased BERT model. We use the transformers’ trainer and default AdamW optimizer with the learning rate setting to $3e^{-5}$ in all the experiments. Training epoch and batch size vary for different datasets (see Table 4). The experiments on AZ-15 and AE-33 takes about 18 hours to run on 4 NVIDIA TITAN XP GPUs. The experiments on AE-48 and AE-110E take less than six hours on single GPU.

	dataset	AE-48	AE-110E	AZ-15	AZ-33
Ask	batch-size	48	48	48	96
	epoch	30	30	5	30
Verify	batch-size	128	128	128	128
	epoch	100	30	10	10

Table 4: batch size and epoch used by Ask-and-Verify on different experiments

The Ask-and-Verify framework has a handful of hyper-parameters. The mentioned values are applicable for all the experiments unless specifically indicated. The maximum number of sub-words for a span candidate is set to 5. The verification model considers at most 5 span candidates in both training and inference stages. Both *span candidate generation* and *verification* input a list of tokens consisting of a text description, a single attribute of interest, and special tokens. The length (*i.e.* number of tokens) of the input sequence K is set to 512. Shorter sentences are padded with a special token. Longer sentences are truncated. In the verification step, the dimension of the span length features d_F is set to 150 and the dropout rate is set to 0.2. The inference threshold τ uses the default value of binary classifier 0.5.

B Preprocessing and Postprocessing

Attribute value extraction is formalized as different tasks by different methods in our experiments, including machine reading comprehension (*i.e.*, question answering), sequence labeling, and *span candidate generation and verification*. There can be multiple ways to conduct preprocessing and postprocessing in these tasks. For example, the labels of MRC can be defined on word-level (before tokenization) or sub-word-level (after tokenization). We tried to ask for the preprocessing code used for the AVEQA (Wang et al., 2020) paper, but were

Product Text Description:	["Stonyfield", "Organic", "Kids", "Whole", "Milk", "Yogurt"]
index:	[0, 1, 2, 3, 4, 5]
Attribute:	"Brand"
Ground Truth:	"Stonyfield Organic"
Span Candidates:	["Stony", "Stonyfield Organic"]
(1) + (2):	"Stony" => "Stonyfield": (0, 0, 1); label: 0 "Stonyfield Organic" => "Stonyfield Organic": (0, 1, 2); label: 1
Tokenized Input Sequence:	["[CLS]", "brand", "[SEP]", "stony", "##field", "organic", ..., "[SEP]"]
index:	[0, 1, 2, 3, 4, 5, ..., 10]
(3):	"Stonyfield": (3, 3, 1); label: 0 "Stonyfield Organic": (3, 5, 2); label: 1

Figure 4: An example of processing span candidates.

unable to obtain it. Following previous sequence labeling work (Zheng et al., 2018; Yan et al., 2021), we first prepare tokens and associated labels in the sequence labeling format for each experiment setting. Then labels of other formats are transformed from the sequence labeling format in the preprocessing step. Output results are later transformed back to sequence labeling format to conduct evaluation. Specifically, we required all the possible extracted attribute values being a subset of continuous full words within the input sequence X for each method. In comparison, standard sub-word tokenizer (*e.g.* BERT) can generate sub-words not necessarily forming full words. For example, if a word is called "swimglass" and sub-word tokenizer can generate "swim" only. We observe the differences have impacts on the evaluation results of different experiment settings.

C Span Candidate Process

In Ask-and-Verify, a span candidate $m_{(s,e)}$ is represented as a tuple (with start-index s , end-index e , span-length $\ell = e - s$). The span candidate must consist of continuous context sub-words with no more than the predefined maximum number of sub-word tokens. Additionally, the span candidates from the generation step are composed of sub-words which do not necessarily form full words. This setting may cause extra errors in final evaluation. Furthermore, the index s and e are not the same as original span candidate’s positions because of sub-word tokenization and the attribute injected in the input sequence. To overcome the interface challenge, we present a processing pipeline in the span candidate collection module to: (1) locate or transform span candidates to nearest tokens forming in the full-word formats; (2) assign binary classification labels with strict string matching between the processed span candidates and ground

Table 5: Models that address the attribute value extraction task and their features on M product attribute types.

	# of models	Scales to AE-110K	Negative Words
OpenTag	M		✓
SuOpenTag	1	✓	✓
AdaTag	2		✓
AVEQA	1	✓	
Ask-and-Verify	1	✓	✓

truth (only in training phase); and (3) capture the correct position of sub-word tokens corresponding to the start token and end token. An example is illustrated in Fig. 4.

D AE-110K dataset

We utilize the public version of AE-110K dataset. However, the data split process, pre-processing, post-processing and evaluation code are not public released. We observed our experiment values for AVEQA (74.6% F1) and SuOpenTag (60.7% F1) are both lower than the values reported in the AVEQA paper: AVEQA (85.01% F1) and SuOpenTag (74.92% F1). This is most likely coming from different data splits and pre-processing strategies.

E Contribution Matrix

In the attribute value extraction task, we argue that there are three major dimensions to judge a framework in the industry production environment: efficiency, scaling ability, and performance. A good framework should have few number of models, capable of scaling up to large number of attributes while obtaining good performances as shown in the Table 5. Compared to all the previous methods, Ask-and-Verify has only one model for multiple attributes, scaling up to thousands of attributes on the public AE-110K dataset, and also carefully considering negative words resulting superior performances on two real word datasets.

F Precision-Recall curve

We present the precision-recall curve of the Ask-and-Verify model on the AZ-15 and AZ-33 settings by changing the threshold values of the Verify step. The performances of other baseline methods are also included in the same figures. From the results in Fig. 5, we can see that the precision and recall keeps a high performance score in a wide range. Compared to the performance of other baselines, the precision and recall curve of Ask-and-Verify

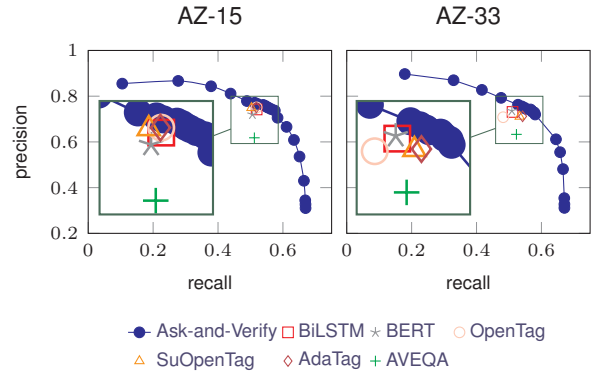


Figure 5: precision and recall curves of Ask-and-Verify on AZ-15 and AZ-33 settings. Ask-and-Verify has a good performance and high flexibility. Both precision-recall curves are above all the comparing methods.

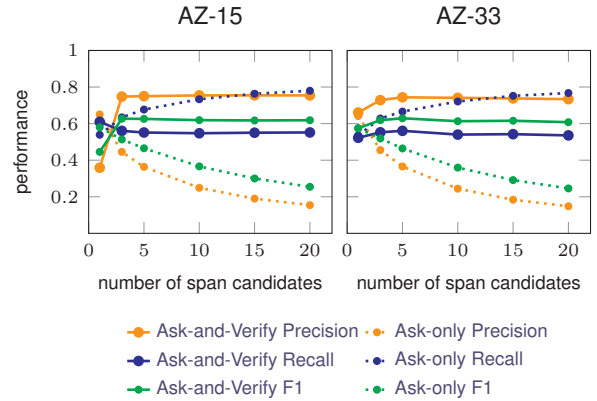


Figure 6: Ablation study on changing the number of span candidates on AZ-15 and AZ-33 settings. With number of span candidates increase, the ask-only provides spans with higher recall but lower precision. In comparison, Ask-and-Verify model has a moderate increase from 1 to 5 span candidates and keeps a stable performance with 5 or more span candidates.

is always on the top right. Interestingly, the margin gets larger on the AZ-33 compared to AZ-15, showing the better scalability of Ask-and-Verify.

G Effectiveness of Changing the Number of Span Candidates

Span candidate plays a central role in the framework by bridging Ask model and Verify model. Intuitively, increasing the number of span candidates can potentially include extra positive span samples but also bring more negative samples at the same time. It is also interesting to investigate how the trade offs impact the performances of overall architecture. Fig. 6 shows the metrics of Ask-only and

Ask-and-Verify models by generating or utilizing the same number of span candidates. We first analyze the Ask-only performances from the dash lines. When the number of span candidates increases, we can see that the recall curve is constantly increasing while precision and f1 curves keep decreasing. Considering the top-20 span candidates, the recall can be even larger than 75% while the precision is only around 25%. As for the Ask-and-Verify performance represented with solid lines, it shows interesting patterns. With top-1 span candidates on AZ-15 setting, it has slightly larger precision and slightly lower recall compared to Ask-only outputs. As the number of span candidates increases to 5, both precision and recall increase. As the number of span candidates keeps increasing, both precision and recall drop slightly but still keep stable scores.

H Case Study

We present some examples to show how Ask-and-Verify span candidate verification step can better capture the correct attribute values from the span candidates. These examples are illustrated in Fig. 7 (A-D). The AVE task of (A) is to extract the “skin tone” attribute from the tanning product. The Ask model produces span candidates with “darker”, “softening and tan extending DHA”, “all” and “black”, sorted by the ranking scores of the Ask model. The “darker” span candidate ranks first and is the output of the Ask-only model. However, the verification model chooses the third-ranked span candidate “all” as the output, matching the ground truth. The AVE task of example (B) is to extract the “hair type” attribute from a Hair Conditioner product. All span candidates are incorrect and Ask-and-Verify is able to reject all the irrelevant attribute values. Example (C) seeks to extract the same “hair type” attribute from a gel product. Even if the correct attribute values are not included within the span candidates, Ask-and-Verify can still reject each candidate and therefore reduce false positives. Finally, in example (D) we seek to extract the “Age Range” attribute from a pack of diapers. Ask-and-Verify can correctly identify the “Baby” attribute value in the second span candidate.

In summary, we find that the Ask-step predicts frequent and contextually-coherent span candidates. However, these span candidates carry many false positives. Introducing the verification-step into the framework appears to substantially reduce the occurrence of false positives.

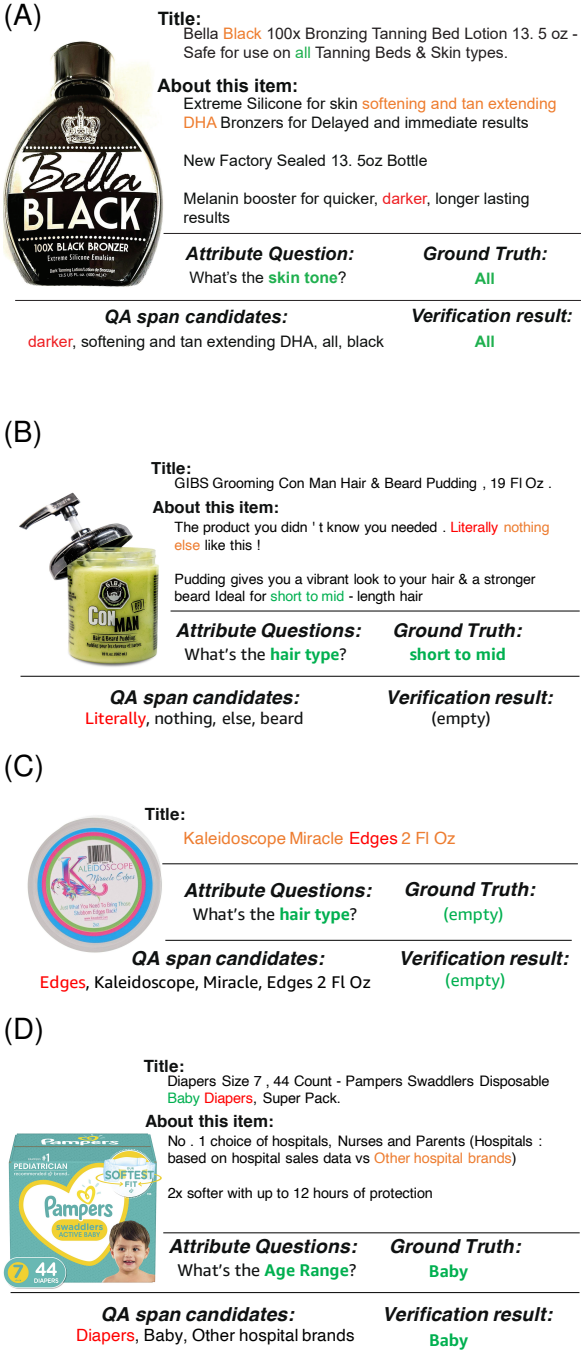


Figure 7: Case study with four illustrative examples of (A) tanning lotion, (B) hair conditioner, (C) hair gel, and (D) diapers. We find that the Ask-step of the Ask-and-Verify model is able to produce reasonable, but noisy candidates each of the product attributes. However, the Verify-step is able to filter-out spurious candidates and reduce the rate of false positives.

Consultation Checklists: Standardising the Human Evaluation of Medical Note Generation

Aleksandar Savkov¹, Francesco Moramarco^{1,2}, Alex Papadopoulos Korfiatis¹, Mark Perera, Anya Belz^{2,3}, Ehud Reiter²

¹Babylon ²University of Aberdeen ³ADAPT Research Centre, Dublin City University

¹ {sasho.savkov, francesco.moramarco, alex.papadopoulos} @babylonhealth.co.uk

² {r01fm20, ehud.reiter, anya.belz}@abdn.ac.uk

Abstract

Evaluating automatically generated text is generally hard due to the inherently subjective nature of many aspects of the output quality. This difficulty is compounded in automatic consultation note generation by differing opinions between medical experts both about which patient statements should be included in generated notes and about their respective importance in arriving at a diagnosis. Previous real-world evaluations of note-generation systems saw substantial disagreement between expert evaluators. In this paper we propose a protocol that aims to increase objectivity by grounding evaluations in Consultation Checklists, which are created in a preliminary step and then used as a common point of reference during quality assessment. We observed good levels of inter-annotator agreement in a first evaluation study using the protocol; further, using Consultation Checklists produced in the study as reference for automatic metrics such as ROUGE or BERTScore improves their correlation with human judgements compared to using the original human note.

1 Introduction

While Electronic Health Record systems are a necessity in modern healthcare, they are burdening primary care clinicians with significant clerical work that distracts them from patient care and increases their dissatisfaction and burnout rates (Arndt et al., 2017). Since a significant part of the required documentation involves note writing, there has been a mounting interest in assisting clinicians by automatically generating consultation notes (Finley et al., 2018; Enarvi et al., 2020; Moleenaar et al., 2020; Knoll et al., 2022).

A common approach involves passing the recording of the consultation through a speech-to-text system, then using a sequence-to-sequence model trained on parallel transcript and note datasets to automatically generate the note (Krishna et al., 2020;

Transcript	Note
Clinician: Hello there, it's Dr Smith, and how can I help you this afternoon?	3/7 hx developed headache. Constant, severity 8/10, dull ache with associated sharp pain, gradual onset.
Patient: Hi there. Well, I have this like really crazy headache that's been going on for days.	Progressively worsening. Has tried ibuprofen with limited relief.
Clinician: Ohh dear, OK. When, when did it exactly start, this headache?	Feels nauseous, no vomit. No neck pain/stiffness. No speech disturbances. No arm or leg weakness.
Patient: Eh, around three days ago, maybe.	No head injury. No fevers. No rashes.
Clinician: Three days ago, OK. And whereabouts in your head, is this pain?	PMH: Nil. DH: Nil. NKDA FH: mother and sister - migraines
Patient: Um, it kind of feels all over my head, but mainly around my right eye. [...]	SH: lives with housemates, works in IT Socially smoke/EtOH.

Table 1: Abridged version of a mock transcript and human-written note from Papadopoulos Korfiatis et al. (2022).

Joshi et al., 2020; Zhang et al., 2021; Moramarco et al., 2022). An example of a transcript and associated consultation note, taken from the mock consultation dataset released by Papadopoulos Korfiatis et al. (2022), can be seen in Figure 1.

Evaluating the output of such systems is challenging (Gehrmann et al., 2022), as it is often the case in Natural Language Generation (NLG). Widely used automatic metrics, such as ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002), often fail to capture relevant aspects of generated text (Reiter and Belz, 2009), and human evaluation, the best practice in NLG, is not only expensive and hard to reproduce (Belz et al., 2021) but also highly subjective (Howcroft et al., 2020; van der Lee et al., 2021; Gehrmann et al., 2022). Even in the field of Note Generation where evaluators tend to be medical experts rather than crowd-sourced workers, inter-annotator agreement is low, as there is no explicit ground truth and the annotators have dif-

fering opinions on the importance of each patient statement and whether it should be included in a consultation note (Moramarco et al., 2022).

In this work, we propose an evaluation protocol that uses Consultation Checklists (CC), itemised reference of all facts discussed during doctor-patient consultations. We report good agreement between clinicians building CCs from the same consultation, which indicates good consistency of the reference creation process. Since Consultation Checklists act as an approximation of the ground truth, they reduce evaluator subjectivity, which is reflected in the high inter-annotator agreement observed in our first study. We also show that correlation with human judgements increases when using CCs instead of the original clinician note as the reference for automatic evaluation metrics.

2 Related Work

There are a number of different approaches to quantitative human evaluation in NLG.

Rating or Likert scales work well with few criteria, but lack explanatory power and fail to capture text quality (Hastie and Belz, 2014). Adding more criteria partly resolves this, but at the cost of the evaluation task becoming more difficult and subjective (van der Lee et al., 2021). For example, Moen et al. (2016) use a 30 item rating scale and report that subjects found it too difficult to use.

Ranking methods, where evaluators are asked to rank the output of text generation systems along a specified criterion, are an alternative to rating scales. Some studies have shown ranking to be more reliable and consistent (van der Lee et al., 2021); however, ranking methods do not scale well when comparing multiple models.

Extrinsic measures, such as measuring post-edit time of generated text (Moramarco et al., 2022) provide a better estimate of how useful the generated text may be to the final user, but are often expensive and subjective (Lai et al., 2022).

A common shortcoming among all methods described above is that none of them provide granular insights into the text generation systems’ errors and how to address them. This is particularly important when evaluating automatically generated medical notes, where the factual accuracy and completeness of the generated note are critical, as well as identifying the situations where the system fails.

As with all summarisation tasks, Note Generation has an element of content selection that is highly subjective. For this reason, evaluating system-generated notes against a single reference summary would penalise those notes that diverge from the reference in their content selection. One way to address this is by using multiple reference summaries¹, as for example in the Pyramid evaluation protocol (Nenkova and Passonneau, 2004). In a similar way to our proposed protocol, it splits the evaluation into two independent steps: extracting Summarization Content Units (SCUs) from multiple references, then using these SCUs to evaluate generated text.

Another way to address the subjectivity of using single references comes from *reference-less* approaches that compare the generated text against the source text directly rather than against reference summaries. For example, Narayan et al. (2019) once more split the evaluation in two steps: highlighting annotations in the source document, then comparing each generated summary to these annotations. In the domain of note generation, however, the format of the source documents (consultation transcripts) and the generated summaries (consultation notes) is different enough (see Table 1) that a highlighting-based approach would not work.

In the medical domain Moramarco et al. (2022) give evaluators the consultation audio recording instead of a reference, and ask them to identify the missing and incorrect items in a generated note, providing the required insight into how the generated text is wrong. However, even though the evaluators are experts (medical practitioners), agreement between them is very low. While this could be improved by better evaluator training, we believe that the evaluation task itself inherently inhibits agreement. There is no standard way of recognising or mapping facts from the audio recording of the consultation, which is the ‘ground truth’, to the consultation note – generated or otherwise. This makes it very hard to align multiple evaluators and get consistent results.

3 Proposed Protocol

We propose Consultation Checklists — a protocol using an expert-crafted ground truth approximation to evaluate the quality of system-generated medical notes with human raters. The evaluation protocol

¹Multiple reference summaries are also used in some automatic evaluation metrics, such as ROUGE (Lin, 2004).

A	B	C	D	E	F	G
Checklist	Importance	Present / Absent		Note	Correct / Incorrect	Importance
PRESENTING COMPLAINT				PC: Pain in the left side of the head.	Y	C
Headache	C	Y		HPC: Onset of a headache for the past day.	Y	C
↳ 1 day	C	Y		Safety Netting: Nil abnormal smells or tastes,	Y	C
↳ since she woke up yesterday morning	C	N		↳ Nil nausea or vomiting.	Y	C
↳ throbbing	C	Y		PC: Headache.	Y	NC
↳ left side of head	C	Y		Factors : light,	Y	C
↳ intermittent	C	N		↳ sensitive to light.	Y	C
↳ sensitive to light	C	N		E&D: Fine.	Y	C
↳ headache still present	C	Y		PC: Left sided headache.	Y	NC
PAST MEDICAL HISTORY				Factors : sunlight,	Y	NC
Nil	C	N		↳ lying flat,	N	NC
DRUG HISTORY				↳ bending forward.	N	C
States allergy to aspirin	C	Y		Meds: None.	N	C
↳ Feels sick when she takes aspirin	C	N		Factors : lying in bed,	N	NC
FAMILY HISTORY				↳ lying flat.	N	NC
Mum - hypertension and migraines	C	Y		Meds : Meds.	N	C
SOCIAL HISTORY				Allergy to aspirin.	Y	C
Lives with friends	NC	N		Safety Netting : Nil changes in the skin,	N	C
Non-smoker	C	N		↳ Nil weakness or numbness in the arms or legs.	Y	C

Figure 1: Columns A and B are the abridged version of a Consultation Checklist. Column E is the automatically-generated note. Columns C, F, and G are filled in by the evaluating clinician.

consists of a reference creation step followed by a notes evaluation step (see Figure 2).

3.1 Creation

Given a dataset of consultation audio recordings, one expert clinician is asked to listen to the audio and produce a Consultation Checklist: a structured list of **all the facts** discussed in the consultation. Including both relevant and irrelevant content in the Consultation Checklist is an important feature of the protocol as it eliminates the subjectivity of the content selection characteristic of other human-made references as discussed in Section 2. The list items are organised in sections for clarity and split into sub-lists to allow for more granularity (e.g. ‘headache for 1 day’ may be item ‘Headache’ with sub-item ‘1 day’); see Columns A and B in Figure 1. Following Moramarco et al. (2022), each item is marked for clinical importance as follows:

Critical: Items medico-legally required to document the diagnosis and treatment decisions whose absence or incorrectness may lead to wrong diagnosis and treatment later on, e.g. the symptom ‘cough’ in a suspected chest infection consultation. This is the key information a note needs to capture correctly in order to not mislead clinicians.

Non-critical: Items that should be documented in a complete note but whose absence will not affect future treatment or diagnosis, e.g. ‘who the patient lives with’ in a consultation about chest infection.

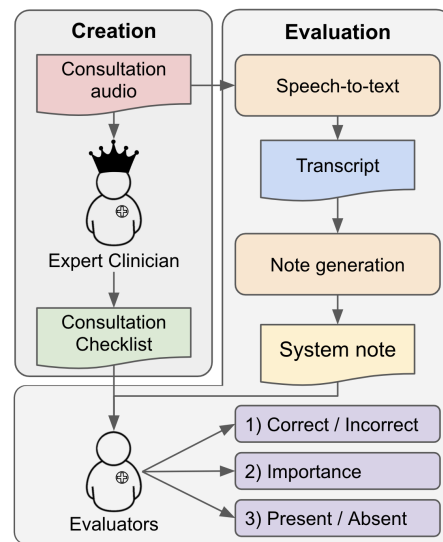


Figure 2: Diagram of the Consultation Checklists evaluation protocol, including the creation stage and the evaluation stage.

Irrelevant: Medically irrelevant information covered in the consultation, e.g. the pet of a patient with a suspected chest infection just died. Including such information in the Consultation Checklist allows for a fair evaluation of the less relevant parts of the generated notes.

Once the Consultation Checklists are created, they can be stored and reused in any future evaluation, thereby making the evaluation cost more scalable.

3.2 Evaluation

The created Consultation Checklists are then used to evaluate one or more system-generated consultation notes by one or more clinicians (or raters). These clinicians are not required to listen to the consultation recording but to only rely on the Consultation Checklist as a common ground when evaluating the notes. Each note is automatically split into sentences, then each sentence is split on punctuation and conjunctions. The first item in each sentence is considered top level and all others are nested in a sub-list (column E in Figure 1). We instruct evaluators to read the full sentence before marking each item as they may be meaningless in isolation. Once familiar with the Consultation Checklist, clinicians are asked to carry out the following sequence of tasks (see Figure 2):

1. Mark each item in the note as *correct* or *incorrect* using the Consultation Checklist as a common reference (column F). Since there is no explicit mapping, the clinicians need to scan the Consultation Checklist to try and find supporting facts for each item in the note. For example, in Figure 1 item 2 in the note ('HPC²: Onset of a headache for the past day') is validated by the first two items in the Consultation Checklist ('Headache' and '1 day').
2. Mark each item in the generated note as *critical*, *non-critical*, or *irrelevant* (column G). Even though some of these values could be inferred from the importance of the Consultation Checklist items, asking the evaluating clinicians to fill them in covers two common edge cases: (i) when one item in the note is fact-checked against multiple items in the Consultation Checklist, and (ii) when the item is incorrect. In the case of incorrect items, the importance is established by the effect the presence of the item may have to the clinician using the generated note.
3. Mark each item in the Consultation Checklist as *present* or *absent* in the generated note (column C) where 'present' means that the item is fully reported in the generated note.

We define *Precision* and *Recall* in the context of Consultation Checklists as:

$$Precision = \frac{|correct\ items|}{|generated\ items|} \quad (1)$$

²History of Presenting Complaint

$$Recall = \frac{|present\ items|}{|checklist\ items|} \quad (2)$$

Both metrics can also be computed for critical items only using the importance level assigned to each item (see Figure 1, column B).

The items in each Consultation Checklist are similar to the 'Summarization Content Units' described by Nenkova and Passonneau (2004) and can be re-used to evaluate any number of generated notes. However, as our method does not use human notes as a reference, the items are extracted from the consultation audio recording rather than from multiple reference summaries. As the evaluation stage involves no writing and only a limited amount of interpretation, the set of clinicians carrying it out could be of lower skill in contrast to the Pyramid approach where the same amount of skill is required for both stages.

4 Checklist Creation Pilot

Based on our initial assumption that a Consultation Checklist should capture the salient points of a consultation in an itemised format, we ran a pilot study with 2 clinicians expert in AI annotation, A & B. The goal of the study was to define best practices for creating Consultation Checklists and to evaluate the consistency of their creation between clinicians.

To investigate the agreement on Checklist creation, we asked the two clinicians to produce Consultation Checklists for the same 10 mock consultations taken from Papadopoulos Korfiatis et al. (2022) (see Figure 3 in the Appendix for an example). In order to quantify the alignment between them, two of the authors checked whether the information of each item in Clinician A's version was present in Clinician B's version, and vice-versa.

Based on this analysis, we define the fact coverage for each Consultation Checklist as the number of matching facts divided by the total number of facts. We found that 93.7% of the items in Clinician A's Consultation Checklists were also present in Clinician B's, and 78.3% the other way around. This may indicate that the two clinicians agree on the basic facts to include but Clinician B tended to add more details. Table 2 shows the values for both annotators, the average, and the agreement computed with Krippendorff's Alpha on the binary values for each statement (present or absent).

As part of the pilot, we refined the following process for creating Consultation Checklists:

Annotator	Checklist A (critical)	Checklist B (critical)
Ann1	94.7% (95.7%)	79.8% (79.1%)
Ann2	92.8% (94.3%)	76.9% (78%)
Avg	93.7% (95%)	78.3% (78.6%)
Agreement	0.624	0.77

Table 2: Results of the alignment between Checklists.

1. Listen to the consultation audio and take notes on every patient statement.
2. Format the notes into an itemised list, splitting longer items to a more atomic level in sub-lists and categorising them using sub-headers (e.g. Presenting History, Past Medical History, Social History, etc.). More examples of sub-headers can be seen in Figure 3 (Appendix A).
3. Read through two system-generated notes to sanity-check the Checklist with regards to item granularity and coverage.
4. Mark each item in the Checklist as critical, non-critical or irrelevant, as defined in Section 3.
5. Re-listen to the consultation audio to ensure no important points have been missed.

The pilot study also highlighted the cognitive effort of producing Consultation Checklists. On average, a Consultation Checklist for a 10 minutes consultation contains 56 items (excluding sub-headings) and it takes the clinician around 1 hour to complete. Also, the clinicians found that they would be able to produce 4 Consultation Checklists in a row before requiring a break, and that the first one would be the quickest to make (in as little as 30 minutes) but the following ones would require progressively more time.

5 Consultation Notes Evaluation

To test the utility of our protocol, we used a single expert clinician to create 20 Consultation Checklists from real-life patient consultations. We then generated History and Examination notes for each of the consultations using a BART (Lewis et al., 2020) encoder-decoder transformer model³. The model was pre-trained on the CNN/Dailymail dataset (Hermann et al., 2015), and fine-tuned on a proprietary dataset of 10,000 (speech-to-text generated) consultation transcripts and human-written History and Examination notes.

³<https://huggingface.co/facebook/bart-large-cnn>

Finally, we hired three clinicians (two women and one man from ethnically diverse backgrounds) to evaluate the 20 generated notes by following the evaluation process defined in Section 3. The training for this task involved: (i) task instructions (see Appendix); (ii) two evaluation practice tasks; (iii) an alignment session between the three clinicians, where they investigated all cases of disagreement on the practice tasks and came to a joint decision.

	Present / Absent	Correct / Incorrect	Imp.
# data points	2258	904	904
Eval 1 - Eval 2	0.733	0.690	0.521
Eval 1 - Eval 3	0.729	0.627	0.387
Eval 2 - Eval 3	0.754	0.701	0.697
3-way Agreement	0.739	0.672	0.522

Table 3: Krippendorff’s Alpha inter-annotator agreement scores for Present/Absent, Correct/Incorrect and Importance (Imp.).

	CC (pairwise)	CC (count)	No CC (count)
Pre / Abs	0.739	0.969	0.374*
Cor / Inc	0.672	0.726	0.541*

Table 4: Krippendorff’s Alpha scores for Present/Absent and Correct/Incorrect values for an evaluation using the Consultation Checklist as common ground and one using the consultation recording. Asterisk(*) denotes scores reported in Moramarco et al. (2022).

In addition, we wanted to quantify how often evaluators misjudge the generated note because of information that is omitted or misrepresented in Consultation Checklists. In order to do this, we asked each evaluator to listen to the audio of the consultation and review whether a generated note item was correct or incorrect.

Metric	Spearman correlation coefficient		Pearson correlation coefficient	
	Ref: Human note	Ref: Checklist	Ref: Human note	Ref: Checklist
Rouge1 Fscore	0.493	0.553	0.509	0.553
Rouge2 Fscore	0.376	0.570	0.445	0.545
Rouge3 Fscore	0.348	0.424	0.408	0.442
RougeL Fscore	0.431	0.636	0.439	0.577
BERTScore	0.375	0.58	0.414	0.563
Levenshtein Distance [†]	0.003	0.284	0.064	0.224

Table 5: Spearman and Pearson correlation coefficients against the human judgements (an average of precision and recall) from the study. [†] denotes lack of statistical significance ($p \geq 0.05$).

6 Results & Discussion

6.1 Inter-Annotator Agreement

One of the advantages of the Consultation Checklists protocol is that it allows us to compute inter-annotator agreement at a item level, as opposed to just error counts as done in [Moramarco et al. \(2022\)](#). Table 3 shows Krippendorff’s Alpha⁴ ([Krippendorff, 2018](#)) scores for the raw pairwise values of *Present / Absent*, *Correct / Incorrect*, and *Importance*. We use nominal agreement for the first two, which have binary values and ordinal agreement for Importance by converting *irrelevant*, *non-critical*, and *critical* to integers.

In order to compare agreement to [Moramarco et al.](#)’s reported results, we also compute interval agreement on error counts. While the results are not directly comparable since the dataset, sample size and annotator count are different, the agreement using the Consultation Checklists protocol is much higher (Table 4). It is also a considerable increase over the average NLG human evaluation agreement of 0.3 to 0.5 reported by [van der Lee et al. \(2021\)](#).

6.2 Accuracy Trade-off

As mentioned in Section 3, when generating Consultation Checklists, the goal is to capture as much of the consultation as possible. However, it is difficult to capture all points while keeping the Consultation Checklist concise, and some nuances which might be needed to faithfully assess the generated notes could be missed.

This trade-off between evaluation accuracy and standardisation is a limitation of our approach that we quantified by checking how often evaluators changed their Correct / Incorrect answers after listening to the consultation audio (Table 6). On average, this only happened for a small number

of generated note items (3.91%). Most changes were from Incorrect to Correct, which highlights the importance of making sure the Consultation Checklists are a thorough representation of the consultation. For example, consider a checklist that includes the item “was feeling cold” but omits the extra information of “had to wear more clothes than usual”. In this case, a generated note item referring to this extra information would be marked as Incorrect based on the Consultation Checklist, but as Correct based on the audio.

Evaluator	# Changes	Correct -> Incorrect	Incorrect -> Correct
Eval 1	44 (4.87%)	9	35
Eval 2	39 (4.31%)	4	35
Eval 3	23 (2.54%)	3	20
Avg	35 (3.91%)	5.33	30

Table 6: Changes in Correct / Incorrect values after listening to the consultation recording.

6.3 Time Efficiency

It took clinicians a self-reported 45 minutes on average to complete each evaluation task: 5 minutes to understand the Consultation Checklist; 15 minutes for each of the two notes to evaluate correct vs. incorrect and present vs. absent items, including item importance; and 10 minutes to listen to the consultation audio and modify any answers. For context, [Moramarco et al. \(2022\)](#) report that their evaluators need 1 hour to listen to a consultation audio recording and evaluate 5 consultation notes.

6.4 Consultation Checklists for automatic evaluation

Since Consultation Checklists aim to be a standardised reference, we expected that their textual

⁴<https://pypi.org/project/krippendorff/>

representation⁵ can also be used as a more objective reference than a single clinician’s consultation note for automatic metrics. To test this, we computed the scores for a few common NLG metrics: ROUGE (Lin, 2004), BERTScore (Zhang et al., 2019) and Levenshtein distance (Levenshtein et al., 1966) using either clinician-written notes or Consultation Checklists as the reference text. Table 5 reports both Spearman and Pearson correlation coefficients against our human judgements (an average of precision and recall). For all three metrics, using Consultation Checklists as references increases the correlation with human judgements, with BERTScore showing the highest gain at 20.5% Spearman’s correlation increase.

7 Conclusion

In this work we proposed a novel reference data structure called Consultation Checklists (CC), and a protocol that uses it for evaluating automatically generated consultation notes. Our experiments show good inter-annotator agreement levels when parallel CCs are created from the same set of clinical consultations. We also report good agreement when the Consultation Checklist protocol is used by different clinicians to evaluate the same consultations. Finally, we showed that expertly-crafted Consultation Checklists are better than human-written notes when used as references for automatic evaluation.

While we have tested our protocol only on note generation for primary care consultations, we postulate that consultation checklists would apply to a number of medical domains, including secondary care and

8 Ethical Considerations

We considered the ethical implications of this work and found no concerns. The study participants are senior clinicians with at least 5-10 years of experience consulting. They are paid £70 an hour, have agreed to work for a maximum of 8 hours per week, and able to withdraw at any time. The consultations they are asked to evaluate are real doctor-patient interactions. These are stored securely following GDPR practices and the patients have consented for their data to be used for research purposes.

Finally, while we hope it can be generalised and applied to other domains, medical and otherwise,

⁵We get the textual representation of a Consultation Checklist by concatenating all items in a single string.

the evaluation protocol we propose in this paper has only been tested in the domain of primary care UK consultations.

References

- Brian G. Arndt, John W. Beasley, Michelle D. Watkinson, Jonathan L. Temte, Wen-Jan Tuan, Christine A. Sinsky, and Valerie J. Gilchrist. 2017. [Tethered to the EHR: Primary Care Physician Workload Assessment Using EHR Event Log Data and Time-Motion Observations](#). *The Annals of Family Medicine*, 15(5):419–426. Publisher: The Annals of Family Medicine Section: Original Research.
- Anja Belz, Anastasia Shimorina, Shubham Agarwal, and Ehud Reiter. 2021. The reprogen shared task on reproducibility of human evaluations in nlg: Overview and results. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 249–258.
- Seppo Enarvi, Marilisa Amoia, Miguel Del-Agua Teba, Brian Delaney, Frank Diehl, Stefan Hahn, Kristina Harris, Liam McGrath, Yue Pan, Joel Pinto, Luca Rubini, Miguel Ruiz, Gagandeep Singh, Fabian Stemmer, Weiyi Sun, Paul Vozila, Thomas Lin, and Ranjani Ramamurthy. 2020. [Generating Medical Reports from Patient-Doctor Conversations Using Sequence-to-Sequence Models](#). In *Proceedings of the First Workshop on Natural Language Processing for Medical Conversations*, pages 22–30, Online. Association for Computational Linguistics.
- Gregory Finley, Erik Edwards, Amanda Robinson, Michael Brenndoerfer, Najmeh Sadoughi, James Fone, Nico Axtmann, Mark Miller, and David Suendermann-Oeft. 2018. [An automated medical scribe for documenting clinical encounters](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–15, New Orleans, Louisiana. Association for Computational Linguistics.
- Sebastian Gehrmann, Elizabeth Clark, and Thibault Selam. 2022. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *arXiv preprint arXiv:2202.06935*.
- Helen Hastie and Anja Belz. 2014. A comparative evaluation methodology for nlg in interactive systems. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14)*, pages 4004–4011.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.
- David M Howcroft, Anja Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A Hasan, Saad Mahamood,

- Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: Nlg needs evaluation sheets and standardised definitions. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 169–182.
- Anirudh Joshi, Namit Katariya, Xavier Amatriain, and Anitha Kannan. 2020. Dr. summarize: Global summarization of medical dialogue by exploiting local structures. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 3755–3763.
- Tom Knoll, Francesco Moramarco, Alex Papadopoulos Korfiatis, Rachel Young, Claudia Ruffini, Mark Perera, Christian Perstl, Ehud Reiter, Anya Belz, and Aleksandar Savkov. 2022. User-driven research of medical note generation software. *In press: NAACL*.
- Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.
- Kundan Krishna, Sopan Khosla, Jeffrey P Bigham, and Zachary C Lipton. 2020. Generating soap notes from doctor-patient conversations. *arXiv preprint arXiv:2005.01795*.
- Vivian Lai, Alison Smith-Renner, Ke Zhang, Ruijia Cheng, Wenjuan Zhang, Joel Tetreault, and Alejandro Jaimes. 2022. An exploration of post-editing effectiveness in text summarization. *In press: NAACL*.
- Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.
- 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.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- Hans Moen, Laura-Maria Peltonen, Juho Heimonen, Antti Airola, Tapio Pahikkala, Tapio Salakoski, and Sanna Salanterä. 2016. [Comparison of automatic summarisation methods for clinical free text notes](#). *Artificial Intelligence in Medicine*, 67:25 – 37.
- Sabine Molenaar, Lientje Maas, Verónica Burriel, Fabiano Dalpiaz, and Sjaak Brinkkemper. 2020. [Medical Dialogue Summarization for Automated Reporting in Healthcare](#). *Advanced Information Systems Engineering Workshops*, 382:76–88.
- Francesco Moramarco, Alex Papadopoulos Korfiatis, Mark Perera, Damir Juric, Jack Flann, Ehud Reiter, Anja Belz, and Aleksandar Savkov. 2022. Human evaluation and correlation with automatic metrics in consultation note generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5739–5754.
- Shashi Narayan, Andreas Vlachos, et al. 2019. Highres: Highlight-based reference-less evaluation of summarization. *arXiv preprint arXiv:1906.01361*.
- Ani Nenkova and Rebecca J Passonneau. 2004. Evaluating content selection in summarization: The pyramid method. In *Proceedings of the human language technology conference of the north american chapter of the association for computational linguistics: Hlt-naacl 2004*, pages 145–152.
- Alex Papadopoulos Korfiatis, Francesco Moramarco, Radmila Sarac, and Aleksandar Savkov. 2022. Primock57: A dataset of primary care mock consultations. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 588–598.
- 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*, pages 311–318.
- Ehud Reiter and Anja Belz. 2009. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4):529–558.
- Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Kraemer. 2021. [Human evaluation of automatically generated text: Current trends and best practice guidelines](#). *Computer Speech Language*, 67:101151.
- Longxiang Zhang, Renato Negrinho, Arindam Ghosh, Vasudevan Jagannathan, Hamid Reza Hassanzadeh, Thomas Schaaf, and Matthew R Gormley. 2021. Leveraging pretrained models for automatic summarization of doctor-patient conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3693–3712.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Appendix

A.1 Evaluating Clinicians instructions

1. Read through the checklist (on the left side of the spreadsheet) and familiarise yourself with the consultation.
2. Based on the information in the checklist, go through the generated note (on the right side of the spreadsheet) and mark each statement of the note as **correct** (y) / **incorrect** (n). [Column G]
3. Mark each statement in the note as either critical, non critical or irrelevant: [Column I]
 - (a) **Critical:** The statement is of critical medical importance. If it's a correct statement, the note would not be medically complete without the statement present; if it's an incorrect statement, its presence in a consultation note would be a medical risk (for example, could lead to a different diagnosis).
 - (b) **Non-critical:** The statement is of medical value, but its presence or absence in the note is not critical medically. For example, some doctors might include a non-critical statement in their note but other doctors could skip it.
 - (c) **Irrelevant:** The statement is irrelevant; if correct, most doctors would not include it in the note (for example, "Patient reports they prefer to wear green clothes"). If incorrect, its presence in the note is inconsequential.
4. Go through each statement in the checklist and mark it as either present or absent in the generated note. [Column C]
5. Repeat this for each generated note
6. Now, listen to the actual consultation recording. Take notes if you need to, especially if something you hear is different from what you understood through reading the consultation checklist.
7. Finally, fill the "Correct / Incorrect (after audio)" field, essentially amending your earlier answers after listening to the consultation audio. This will allow us to evaluate how much information and context is lost by using the consultation checklist instead of the actual recording. [Columns D,I]

A.1.1 Checklists comparison

Checklist (doctor 1)	Importance	Checklist (doctor 2)	Importance
SETTING		INTRODUCTION	
Patient confirmed he was in a private place to speak	C	Doctor's credentials	NC
PRESENTING HISTORY		Confirm name	NC
Elbow symptoms	C	Confirm DOB	NC
↳ Left elbow	C	In a private place	NC
↳ 1 week history	C	Able to speak freely	NC
↳ Pulling sensation	C	PRESENTING COMPLAINT	
↳ Initially noted while patient was in the shower	C	Problem with elbow	C
↳ Patient feels like there maybe "some fluid"	C	↳ Left	C
↳ Patient reports he does not know what is going on	NC	↳ Swelling	C
↳ Slightly warm.	C	↳ 1 week	C
↳ Not painful.	C	↳ Not painful	C
BACKGROUND HISTORY		↳ Noticed when in the shower	I
No previous injury	C	↳ Slightly warm	C
Works at a desk	C	↳ Feels fluid like inside	C
↳ not a physical job	NC	↳ No injury	C
No previous episodes	C	↳ Feels strange	NC
PAST MEDICAL HISTORY		PAST MEDICAL HISTORY	
possible osteoarthritis	C	Not happened before	C
↳ reportedly confirmed by a GP	C	Thinks has OA - diagnosed by GP previously	C
↳ Confirmed a few years ago.	C	No eczema	I
No known other diagnoses.	C	DRUG HISTORY	
No eczema.	NC	No known drug allergies	C
FAMILY HISTORY		Nil meds	C
no known family history of rheumatoid arthritis	C	Allergic to peanuts	I
no known family history of lupus		FAMILY HISTORY	
DRUG HISTORY		Unsure if hx of RA or lupus	C
↳ nil regular.	C	SOCIAL HISTORY	
↳ Allergic to peanuts.	NC	Sits at a desk	C
↳ No known drug allergies	C	Runs regularly 2/3 times a week	NC
SOCIAL HISTORY		EXAMINATION	
↳ Runs 2-3 times/week.	C	Tip of elbow is warm	C
↳ Patient considers himself healthy.	NC	Skin is dry on tip of elbow	C
DIGITAL EXAMINATION:			
Doctor asked patient to demonstrate some movements/actions	C		
↳ Some pain around where the swelling is	C		
↳ Skin was noted to have been dry in this region	C		

Figure 3: Example of two checklists for the same mock consultation.

Towards Need-Based Spoken Language Understanding Model Updates: What Have We Learned?

Quynh Do*
Amazon Alexa AI
doquynh@amazon.de

Judith Gaspers*
Amazon Alexa AI
gaspers@amazon.de

Daniil Sorokin*
Amazon Alexa AI
dsorokin@amazon.de

Patrick Lehnen
Amazon Alexa AI
plehnen@amazon.de

Abstract

In productionized machine learning systems, online model performance is known to deteriorate over time when there is a distributional drift between offline training and online application data. As a remedy, models are typically retrained at fixed time intervals, implying high computational and manual costs. This work aims at decreasing such costs in productionized, large-scale Spoken Language Understanding systems. In particular, we develop a need-based re-training strategy guided by an efficient drift detector and discuss the arising challenges including system complexity, overlapping model releases, observation limitation and the absence of annotated resources at runtime. We present empirical results on historical data and confirm the utility of our design decisions via an online A/B experiment.

1 Introduction

Traditionally, a Spoken Language Understanding (SLU) system like Google Assistant, Siri or Alexa, is a cascade of an Automatic Speech Recognition (ASR) component converting speech into text followed by a Natural Language Understanding (NLU) component that interprets the meaning of the text through domain classification (DC), intent classification (IC) and slot filling (SF). Once deployed to customers, the machine learning (ML) models implemented for DC, IC and SF may experience distributional drifts between offline training and online application data over time which leads to serious performance degrades. This is known as a *model drift* phenomenon. In most real-world cases, model drift can be avoided by retraining the ML models with regular cadence. However, this strategy often entails a significant amount of human, computational and environmental costs in each release cycle, especially for large systems. Hence, it should be optimized with an intelligent decision

making mechanism that is able to automatically predict if a particular model needs to be updated or can be left in place as is for another release cycle.

Given a deployed ML model, *drift detection* is a task to identify model drifts when the model is applied on a new data set, and therefore can be used to guide decisions on when to retrain the model. However, academic work in this field often makes simplifying assumptions that render the problem more manageable but do not hold in the industrial practice, e.g., that there is exactly one model that needs to be kept or updated, and that it is possible to create non-overlapping detection and reference data windows, which are aligned with a model release cycle. This makes productionizing drift detection nontrivial.

This paper describes our effort to develop a drift detector to produce decisions whether to update the NLU models in the next release cycle for a productionized SLU system. First, we generally discuss the challenges that may arise when applying drift detection on a large-scale SLU production system: i) the complexity when the SLU system is supporting multiple domains with several ML models and architectures; ii) the possible overlapping model releases in production; iii) the limited number of observations for each individual ML model; iv) the absence of annotated SLU data at runtime. Moreover, for each challenge, we consider the necessary system design decisions and possible solutions that are needed to apply drift detection in practice. Finally, we describe our offline and online experiments on real-world SLU data to confirm the utility of our design decisions.

2 Background on drift detection

Lu et al. (2018) classifies automatic drift detection methods into three categories: i) methods which rely on labeled data to monitor error rates, ii) methods which use distance measures to estimate the similarity between distributions of previous and

*These authors contributed equally to this work

current (unlabeled) data, and iii) methods that make use of multiple hypothesis tests to detect concept change. While the first category requires manually labeled data representing the current data distribution, the last two categories require at least two data windows: a *reference window* containing the instances that belong to the same distribution that was used to train the most recent model, and a *detection window* which represents the current data distribution. The detection window can consist of unlabeled data only (Gemaque et al., 2020). Thus, the methods of the second and the third categories can be both classified as unsupervised drift detection methods (Elsahar and Gallé, 2019; Qin et al., 2021). For example, Koh (2016) compare the reference and detection data windows by Hoeffding bound. The difference in terms of sample means between both the windows is compared to a value ϵ defined by the Hoeffding bound. Then, a drift is signaled when this difference is greater than ϵ . An obvious advantage of unsupervised methods is that they do not require labeled data. However, it can be difficult to interpret the drift signal due to the lack of an indication on how much the performance drop is.

While drift detection is considered as one of the stages in modern AI workflows, corresponding work in NLP and SLU has been limited. Recently, Do et al. (2021) have proposed a regression model to detect temporal performance drop in SLU. The authors evaluated their approach via small-scale simulated release cycles for a joint IC+SF model in isolation, thus abstracting away from the complexity of a production SLU system. They built one regression model per domain, assuming the availability of a large number of previous model releases for training the regression model, which is unrealistic for many real-world scenarios.

3 NLU drift detection definition

In this work, we consider multi-domain NLU as one part of a more complex production SLU system, leaving out ASR and other components. The main NLU tasks include DC, IC and SF, and there are different ways to approach them. Usually, pipelined systems are constructed with DC being applied as the first step to determine the domain for a given utterance. Subsequently, the utterance is fed into the corresponding domain-specific IC+SF model that jointly detects the intent and extracts semantic slots from the utterance. For instance, an

ASR transcribed utterance “play Hello by Adele” can be parsed into {domain: Music, intent: “play”, song name (slot): “Hello”, artist (slot): “Adele”}.

To simplify the problem, we focus on building a drift detector to decide on IC+SF model updates. In our experiments, we make *an assumption* that DC models also face distributional drifts when their corresponding IC+SF models do, since their data ages are usually similar.

Given a trained IC+SF model \mathcal{M} , a data *reference window*, \mathcal{W}_{ref} , containing the testing instances considered belonging to the data distribution at the time \mathcal{M} was developed offline, and a *detection window*, \mathcal{W}_{detect} , which contains the testing utterances representing the data distribution when the model update decision needs to be made. Then, we define that \mathcal{M} has suffered a drift if the error rate on the detection window exceeds the error rate on the reference window by a certain threshold:

$$\Delta_E = E(\mathcal{M}, \mathcal{W}_{detect}) - E(\mathcal{M}, \mathcal{W}_{ref}) > \alpha \quad (1)$$

where α is a drift threshold, and E is a pre-defined function to compute an error rate. In this work, E is a semantic error rate and defined as follows:

$$SemER = \frac{\#(\text{slot+intent errors})}{\#\text{slots in reference} + 1} \quad (2)$$

A drift detector should be able to identify automatically whether \mathcal{M} has suffered a drift or not.

4 NLU system and challenges

In this section, we describe the considered NLU system, and discuss the potential challenges arising when applying drift detection on such a system.

4.1 NLU architecture and Challenge 1 - system complexity

In this work, we consider a real-world SLU system with multiple domains and each of them has a single IC+SF model. Each IC+SF model is a combination of a pre-trained encoder and two separate decoders for the target tasks. Depending on its domain, the IC+SF model may receive gazetteers as an additional token-level input in parallel to the BERT-encoder embeddings, resulting in two variants of IC+SF model architectures. In the rest of this paper, we refer the gazetteer-based and non-gazetteer architectures as *Gaz* and *Non-Gaz*, respectively.

Traditionally, academic work on drift detection often assumes that there is exactly one model that

needs to be kept or updated to make the problem more manageable. Unfortunately, it does not hold in industrial practice. As in our case, a real-world SLU system is often multi-domain. Each domain IC+SF model can be updated individually and there is no requirement for these models to have the same architecture. Thus, we face the first challenge: A separate drift detector should be developed per domain and architecture or a single detector needs to cover all supported domains.

4.2 NLU lifecycle and Challenge 2 - overlapping model releases

Figure 1 presents a simplified SLU model production lifecycle, where each release has three main phases: Build, Deploy and Production. Let us apply the drift detection definition (Sec. 3) to the lifecycle of a single IC+SF domain model. For IC+SF model \mathcal{M}_N^D that was released for a domain \mathcal{D} during a release cycle N , the task is to predict if there is a drift after \mathcal{M}_N^D was deployed (that is if $\Delta_E > \alpha$) using the data window \mathcal{W}_{ref} collected during the model build and the window \mathcal{W}_{detect} collected after the deployment. Consequently, the drift detection decision indicates if \mathcal{M}_N^D needs to be updated during the release cycle $N + 1$. Thus, the detection window \mathcal{W}_{detect} must be available before the build phase of release $N + 1$, but after the deployment of N .

However, in practice, the build, deploy and production phases of subsequent releases may overlap significantly which make defining disjoint \mathcal{W}_{ref} and \mathcal{W}_{detect} for releases N and $N + 1$ our second challenge. For a complex and large-scale production system, the required time for each phase is considerable and the human and computational cost of each phase are usually distributed between teams. Once the model development team is finished with the latest model build, it is handed over for deployment and the team can start the work on the next release. Therefore, overlaps tend to occur between the cycles of two consecutive releases. In Figure 1, the Build phase of release $N + 1$ starts before the model of release N goes to production. In this case, the data from the detection window for the current release after its deployment is not yet available at the start of the build phase for the next release. Thus, \mathcal{W}_{detect} is not available for application of a drift detector.

To overcome this challenge, we take that the online data collected after the development of a

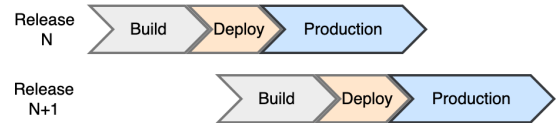


Figure 1: A simplified NLU lifecycle with three phases per release: Build, Deploy and Production. The phases overlap for subsequent releases and the development for the next release $N + 1$ may start as soon as the previous release is deployed.

model for a release N during its build phase is indicative of the drift that might occur after the model go to the deployment stage.

Since the build phase of a model is sufficiently long so that new data can be collected that didn't go into the model development, we will use this data for \mathcal{W}_{detect} . We adjust the definitions for the reference and detection windows as follows:

- *Reference window*: De-identified online data that was manually transcribed and labeled with domain, intent and slots and that was collected before the build phase of a release. This data is used for testing during the build phase. We use the annotated domain labels to split that data between IC+SF domain models. The gold domain labels are used to decide the data flow for each domain's IC+SF model.
- *Detection window*: De-identified online data that was automatically transcribed and automatically classified into domains with the current release DC model and that was collected after the Build phase of N has started, but before the Build phase of $N + 1$. We use the automatically generated domain labels to filter the data for a specific domain.

These definitions result in a time gap between the defined detection window and the real online window, during which the IC+SF model will be deployed. We evaluate this decision in an online experiment in Section 7.

4.3 Challenge 3 - limited observations

The drift detection problem is domain dependent and different domains may experience different drift patterns. For example, the Video domain should observe large drifts more often than the Calendar domain. That would call for learning a separate drift detection function for each domain in production. To learn a drift detector, we need to collect historical data of model releases and reference and detection windows. And to learn a domain

specific drift detector the same data needs to be collected per domain, leading to the third challenge:

For a real-world SLU system, only a very limited amount of available historical data points may be available for training per domain. The availability of historical data for learning a drift detector is restricted by the age of the production system and privacy guidelines for data and model storage. This implies that a per-domain drift detector needs to be learned on a handful of data points, which is often infeasible.

To avoid this issue, we build a single drift detector for all IC+SF domain models of the same type (Gaz or Non-Gaz). We evaluate a single unified drift detection function for multiple domains in the next sections in offline experiments.

4.4 Challenge 4 - the absence of annotated data at runtime

The amount of unlabeled live data flowing into a production SLU system may be on the order of a million of utterances per day. However, due to the associated costs, only a comparatively small subset gets manually transcribed and annotated, leading to the fourth challenge: Manually labeled data for any given period in time may be limited, with potentially only few – or in rare cases even none – manually labeled utterances being available for low-frequency domains in certain time periods. In addition, since manually transcribing and annotating data takes time, this data may not be available at runtime to construct the detection window. As mentioned in the previous section, we use the ASR and DC components to obtain textual per-domain data. The amount of data instances per window can be huge in production, which makes data processing a challenge and may slow down the process. As drift detection should enable quick decisions, we downsample the data amounts to a reasonable size. Since we cannot rely on manually annotated data to extract the features for the detection window, we focus on methods that require only unlabeled data at runtime.

5 Supervised drift detection from unsupervised signals

5.1 Learning problem definition

Motivated by recent work, which successfully predicts a performance drop using unsupervised signals (Elsahar and Gallé, 2019; Do et al., 2021), we aim to learn a function to map from a set of

measures estimating the similarity between the reference and detection data windows to a binary label indicating whether a significant performance drop occurs or not. By predicting directly if a model drift and performance drop occurred instead of estimating a magnitude of the drift, we simplify the learning problem, so that it can be approached with only a limited amount of training data points available. At the same time, the drift detector predictions remain clearly interpretable for the user.

More formally, we learn a function predicting if there is a performance drop when the test data window for model \mathcal{M} trained on \mathcal{W}_{train} is moved from \mathcal{W}_{ref} to \mathcal{W}_{detect} :

$$F(f_1(x), f_2(x), \dots) \rightarrow \{Drop, No-Drop\} \quad (3)$$

where f_1, f_2, \dots are features, x is a data instance containing the information about \mathcal{M} , \mathcal{W}_{train} , \mathcal{W}_{ref} and \mathcal{W}_{detect} . The *No-Drop* label indicates that $\Delta_{SemER} \leq \alpha$ while the *Drop* label indicates that $\Delta_{SemER} > \alpha$.

5.2 Features and learning algorithms

For f_1, f_2, \dots in Equation 3, we explore 17 features representing the differences between \mathcal{W}_{ref} and \mathcal{W}_{detect} as follows:

Discriminative classifier: Discriminative classifiers have been used for drift detection (Elsahar and Gallé, 2019; Do et al., 2021). When \mathcal{W}_{ref} and \mathcal{W}_{detect} are separable by a discriminative classifier, it is likely that there is a drift. In this work, we apply a Logistic Regression classifier trained on the pre-trained BERT sentence representations as the discriminative classifier, and its 5-fold cross-validation accuracy on each of the unions of $(\mathcal{W}_{ref}, \mathcal{W}_{detect})$, $(\mathcal{W}_{ref}, \mathcal{W}_{train})$, and $(\mathcal{W}_{detect}, \mathcal{W}_{train})$ is used as the drift signal.

Distributional distance: For each of the window pairs $(\mathcal{W}_{ref}, \mathcal{W}_{detect})$, $(\mathcal{W}_{ref}, \mathcal{W}_{train})$, and $(\mathcal{W}_{detect}, \mathcal{W}_{train})$, we compute Euclidean and Cosine distances between two mean pre-trained BERT sentence representations.

Prediction confidence differences: Confidence scores have been proven to be effective in detecting drifts (Do et al., 2021). We use the SF and IC prediction confidence scores separately. For each case, we compute the difference between the mean confidence scores that model \mathcal{M} obtains on the two data windows. We also compute the average, minimum and maximum difference between the per tag mean IC confidence scores on the two data windows \mathcal{W}_{ref} and \mathcal{W}_{detect} .

Kullback-Leibler (KL) divergence at token level: KL divergence measures the difference between two probability distributions and is also known as the relative entropy. It has been used as an effective measure in drift detection (Lindstrom et al., 2011). We consider each data window as a large text and compute the KL divergence between the token distributions of the two texts representing each of the data window pairs among \mathcal{W}_{ref} , \mathcal{W}_{detect} and \mathcal{W}_{train} .

Targeting fast drift detection, to learn F we apply classical binary classification algorithms like *multinomial logistic regression*, *k-nearest neighbors*, or *decision tree*. In some cases, it may be desired to reduce the amount of features, e.g., when we have a limited amount of training data points. Therefore, we include feature selection methods into our experiments.

6 Offline experiment

We evaluate the proposed feature set and the selected learning algorithms for Gaz and Non-Gaz architectures using historic data.

6.1 Experimental set-up

In offline experiments, we build two drift detection models for two different IC+SF architectures: i) *Gaz*: Including domains which use gazetteers as an extra input. ii) *Non-Gaz*: Including domains which do not use gazetteers. Following previous work (Do et al., 2021), we define the baseline and metrics as follows. Given N NLU models and historic performance data, a subset $S \subseteq N$ are the models that have no drift (that is, on historic data windows $\Delta_{SemER} \leq \alpha$) and thus belong to the class *No-Drop*. Using the learned drift detector F , $P \subseteq N$ models are predicted as low-risk of being drifted, i.e. predicted as *No-Drop*. To evaluate F , we compute precision: $P = \frac{|S \cap P|}{|P|}$ and recall: $R = \frac{|S \cap P|}{|S|}$ for the *No-Drop* class.

We consider precision for the class *No-Drop* as the most important metric in a user-facing setting. This class includes the models that can be left in place for the next release, thus having the potential to negatively impact customer satisfaction. If a drifted model is left in place, then there is the risk of increased friction for the customers. Recall does not have an impact on customer satisfaction, but on training costs, which we consider less important than customer satisfaction. We aim to reduce training costs without negatively impacting customers.

α	B_p	Model	Feat.	R	P
0.0	45.0	MLR	ALL	55.6	56.8
		Knn	ALL	46.7	55.3
		DT	ALL	57.8	56.5
		MLR	SUB	55.6	75.8
		Knn	SUB	68.9	70.5
		DT	SUB	57.8	68.4
0.002	46.0	MLR	ALL	56.5	61.9
		Knn	ALL	43.5	54.1
		DT	ALL	60.9	56.0
		MLR	SUB	58.7	75.0
		Knn	SUB	56.5	59.1
		DT	SUB	58.7	61.4
0.01	59.0	MLR	ALL	72.9	75.4
		Knn	ALL	66.1	70.9
		DT	ALL	66.1	72.2
		MLR	SUB	84.7	73.5
		Knn	SUB	72.9	72.9
		DT	SUB	83.1	75.4

Table 1: Evaluation results for non-Gaz drift detection model on class No-Drop. B_p , R, P are the precision baseline, Recall and Precision, respectively.

Therefore, our goal is to build a drift detector which reaches a high precision and an acceptable recall for the drift class *No-Drop*. We compare our models against a baseline (B_p) obtained by selecting instances for the *No-Drop* class randomly.

We collected past NLU model release data points resulting in 134 instances for *Gaz* and 100 instances for *Non-Gaz* model architectures. For each release model instance, the utterances representing \mathcal{W}_{ref} and \mathcal{W}_{detect} windows are sampled. All data used in our experiments was de-identified.

We compare the performance of three binary classifiers to learn F : multinomial logistic regression (MLR), k-nearest neighbors (Knn) and decision tree (DT). The classifiers are built using all features (ALL) defined in Section 5.1 or a selected subset of features (SUB) obtained with correlation-based feature selection (Hall, 1999). For training classifiers and feature selection, we use scikit-learn (Pedregosa et al., 2011). We report 10-fold cross-validation performance for the *No-Drop* class with three different drift thresholds α , used to assign the gold labels based on the historic data: 0.0, 0.02, and 0.1.

6.2 Results

Table 1 and Table 2 show the offline evaluation results for the two model architecture types. In most cases, drift detection models outperform the precision baseline. MLR reaches the best precision of 75.8% to select models to be left in place for the next release on non-gazetteer data instances ($\alpha =$

α	B_p	Model	Feat.	R	P
0.0	47.01	MLR	ALL	50.8	58.2
		Knn	ALL	38.1	45.3
		DT	ALL	68.3	59.7
		MLR	SUB	49.2	72.1
		Knn	SUB	50.8	59.3
		DT	SUB	69.8	65.7
0.002	50.75	MLR	ALL	63.2	58.1
		Knn	ALL	42.6	49.2
		DT	ALL	50.0	65.4
		MLR	SUB	76.5	64.2
		Knn	SUB	60.3	64.1
		DT	SUB	61.8	64.6
0.01	61.19	MLR	ALL	78.0	61.0
		Knn	ALL	48.8	58.0
		DT	ALL	85.4	71.4
		MLR	SUB	93.9	65.8
		Knn	SUB	59.8	68.1
		DT	SUB	80.5	72.5

Table 2: Evaluation results for Gaz drift detection model on class No-Drop. B_p , R, P are the precision baseline, Recall and Precision, respectively.

0.0) compared to the random baseline precision of 45.0%. There seems to be no benefit to use a higher threshold to create the gold Drop/No-Drop labels to train a drift detector, so we set α to 0.0 in the online experiment.

Feature selection is often useful in boosting the drift detection performance. From 17 features, depending on the dataset, 1 to 4 features were normally selected. The following features were selected at least once: The Euclidean distance between the mean pre-trained BERT sentence representations, the difference between two mean SF confidence scores, the difference between two mean IC confidence scores, and discriminative classification score. Among these features, the difference between the two mean SF confidence scores seems to be the most important feature, as it is selected in all cases.

7 Online A/B experiment

We conducted an online A/B experiment to evaluate our drift detection approach in the context of a complex large-scale production SLU system (i.e., including several components in addition to NLU, such as ASR). We picked a point in time in which all domain NLU models were scheduled for re-training and re-deployment. The production system with all models updated served as the A model. Our B system comprises exactly the same components, except that we re-trained only a subset of the domain NLU models, according to the decision of our drift detector. We acknowledge that

it would be desirable to include another baseline model with none of the domain models updated into the comparison. Yet, simply leaving all domain models in place as they are increases the risk of model drift, thus potentially increasing friction for the customers being exposed to such a system.

We generated the features for each domain IC+SF model and applied our MLR drift detection model with feature selection and drift threshold $\alpha = 0.0$ (see Section 6). A low threshold for the SemER drop also reduces the risk of negatively impacting customers. If the predicted class was *No-Drop*, the IC+SF model for the domain was left in place, otherwise it was retrained on current traffic. Roughly half of the domain models were re-trained vs left in place. In this work, we assumed that a DC model faces model drift simultaneously with its corresponding IC+SF model as their data ages were usually similar. Therefore, the DC model also followed the same retraining decision as its corresponding IC+SF unless there was a special circumstance like the appearance of a new domain in the next release. Both, the A and the B systems were deployed, and we ran our experiment over 10 days.

By applying a need-based approach to domain model re-training in the B system, our main goal is to decrease costs for model re-training compared to the A system (in which all domain models are re-trained), while keeping model performance the same. To measure the impact on performance, we have monitored online friction metrics that reflect the overall end-to-end system performance (as opposed to the NLU only component in the offline experiments). Unlike in the offline experiments, all metrics were computed automatically and concern overall system performance rather than NLU in isolation. We compared the A and B system’s performances using the selected online metrics, indicating that there was no significant increase recorded in any of the error metrics for the B system compared to the A system. Thus, we conclude that even though we left around half of the domain models in place according to the decision of our drift detector, there was no negative effect on overall system performance. Due to re-training fewer models, we observed a decrease in costs for expensive GPU instances of 46.4% for training IC+SF models (no GPUs were used for detector building).

8 Conclusions

We presented an efficient drift detection approach to guide IC+SF model retraining decisions, which requires only unlabeled data during the application phase in a multi-domain large-scale SLU system. We discussed the challenges that we faced while developing the approach and the corresponding design decisions to address them. We presented experimental results using historical data and we evaluated our approach via both offline and online experiments with a large-scale SLU system, confirming the utility of our design decisions.

Acknowledgements

We would like to thank Yannick Versley, Caglar Tirkaz, Tobias Falke and Debjit Paul for valuable feedback on this work.

References

- Quynh Do, Judith Gaspers, Daniil Sorokin, and Patrick Lehnen. 2021. Predicting temporal performance drop of deployed production spoken language understanding models. In *Proc. Interspeech*.
- Hady Elsahar and Matthias Gallé. 2019. [To annotate or not? predicting performance drop under domain shift](#). 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 2163–2173, Hong Kong, China. Association for Computational Linguistics.
- Rosana Noronha Gemaque, Albert Franca Josua Costa, Rafael Giusti, and Eulanda Miranda dos Santos. 2020. An overview of unsupervised drift detection methods. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10.
- Mark A. Hall. 1999. *Correlation-based Feature Selection for Machine Learning*. Ph.D. thesis.
- Yun Sing Koh. 2016. Cd-tds: Change detection in transactional data streams for frequent pattern mining. *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1554–1561.
- Patrick Lindstrom, Brian Mac Namee, and Sarah Jane Delany. 2011. [Drift detection using uncertainty distribution divergence](#). In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 604–608.
- Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. 2018. [Learning under concept drift: A review](#). *IEEE Transactions on Knowledge and Data Engineering*, page 1–1.
- 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.
- Libo Qin, Tianbao Xie, Wanxiang Che, and Ting Liu. 2021. [A survey on spoken language understanding: Recent advances and new frontiers](#).

Knowledge Distillation Transfer Sets and their Impact on Downstream NLU Tasks

Charith Peris *

Amazon, Cambridge, USA
perisc@amazon.com

Lizhen Tan

Amazon, Cambridge, USA
ltn@amazon.com

Thomas Gueudre

Amazon, Turin, Italy
tgueudre@amazon.it

Turan Gojayevev

Amazon, Berlin, Germany
tgojayevev@amazon.de

Pan Wei

Amazon, Cambridge, USA
panwei@amazon.com

Gokmen Oz

Amazon, Cambridge, USA
ogokmen@amazon.com

Abstract

Teacher-student knowledge distillation is a popular technique for compressing today’s prevailing large language models into manageable sizes that fit low-latency downstream applications. Both the teacher and the choice of transfer set used for distillation are crucial ingredients in creating a high quality student. Yet, the generic corpora used to pre-train the teacher and the corpora associated with the downstream target domain are often significantly different, which raises a natural question: *should the student be distilled over the generic corpora, so as to learn from high-quality teacher predictions, or over the downstream task corpora to align with finetuning?* Our study investigates this trade-off using Domain Classification (DC) and Intent Classification/Named Entity Recognition (ICNER) as downstream tasks. We distill several multilingual students from a larger multilingual LM with varying proportions of generic and task-specific datasets, and report their performance after finetuning on DC and ICNER. We observe significant improvements across tasks and test sets when only task-specific corpora is used. We also report on how the impact of adding task-specific data to the transfer set correlates with the similarity between generic and task-specific data. Our results clearly indicate that, while distillation from a generic LM benefits downstream tasks, students learn better using target domain data even if it comes at the price of noisier teacher predictions. In other words, target domain data still trumps teacher knowledge.

1 Introduction

In the recent past, large language models (LMs; BERT-Large, Devlin et al., 2019; GPT-2, Radford et al., 2019; T5, Raffel et al., 2020) pretrained in a self-supervised manner on massive web corpora have consistently shown state-of-the-art per-

formance for multiple natural language understanding (NLU) tasks. Therefore, it is no surprise that these models are of much interest for virtual assistants such as Amazon Alexa, Apple Siri, and Google Assistant. Some studies have shown that these large models trained on generic corpora seem to be more robust to data distributional shifts, relying less on domain-specific training data to perform well (Brown et al., 2020).

Since large models cannot be directly used for low-latency applications on devices with limited computing capacity, many techniques have been developed to compress them in size. Knowledge distillation (referred to simply as distillation hereafter; Hinton et al., 2015), has shown promising results, especially at the high compression rates typically required in NLU (Jiao et al., 2020, Soltau et al., 2021). In this paradigm, lightweight models referred to as students, are trained to mimic the teacher predictions over a transfer set (Hinton et al., 2015). When the pretraining and task-specific corpora have significantly different distributions, as is often the case, the choice of data for the transfer set can be ambiguous. On the one hand, using pretraining corpora in the transfer set ensures high quality teacher predictions that are important for effective distillation. On the other, using the downstream corpora, although it might cause noisier teacher predictions, ensures the adaptation of the student to its final use case.

To investigate this trade-off, we present a set of experiments where we distill several multilingual students from a large multilingual teacher LM trained using a masked language modeling (MLM) objective. We perform the distillations using transfer sets that comprise of generic and task-specific data in varying proportions. The students are then finetuned and evaluated on two downstream NLU tasks of interest: a Domain Classification (DC) task and a joint Intent Classification/Named Entity Recognition (ICNER) task. For each input utter-

* Corresponding Author

ance DC predicts the relevant domain (Books, Music, Shopping, etc.), IC identifies the user’s intent (find a book, play a song, buy an item, etc.) and NER extracts the entities in the utterance (dates, names, locations, etc.).

Our contributions: (1) We confirm for our setup that model preparation via distillation from a larger LM is more beneficial for downstream task performance when compared to encoder training from scratch. (2) We show that the largest improvements are seen when using only the downstream task’s unlabelled data during the distillation process. Even though teacher predictions are expected to be noisy over data that is different from pre-training corpora, our results clearly indicate that students learn best in this setting. (3) Because our ICNER corpora is divided per domain, we are also able to provide a finer-grained analysis of the impact of corpora similarity on downstream results. (4) Finally, we also confirm that further adaptation of the teacher to the target-domain data, results in improved student performance across tasks.

2 Relevant Work

Building models with inference speeds that are suitable for production systems is of utmost importance in the industrial setting. Therefore techniques for model compression (quantization [Gong et al., 2014](#); pruning redundant connections [Han et al., 2015](#)) have been active research topics, with distillation ([Romero et al., 2015](#), [Hinton et al., 2015](#), [Jiao et al., 2020](#)) showing much promise for NLU models ([Sanh et al., 2019](#)). Distillation processes and their data have evolved over the past few years. In the teacher-student framework proposed by [Hinton et al. \(2015\)](#), they recommend using the original pretraining set as the transfer set. [Jiao et al. \(2020\)](#) proposes a more complex two-stage process with generic and task-specific distillation phases, each with their own data sets, designed to augment the performance of the final model towards the task at hand.

Our work is focused on exploring how varying proportions of generic and task-specific data within the transfer set of a single distillation process impacts downstream NLU performance. Since our scope does not include optimizing the distillation process itself, we use a cheaper alternative to [Jiao et al. \(2020\)](#), via a single-stage distillation setup to conduct our exploration (see Section [A.3](#) for details).

[Gururangan et al. \(2020\)](#) showed for the *pretraining* phase, that continued domain-adaptive and task-adaptive pretraining using the downstream task’s unlabeled data can improve performance. Our work presents similar results for the distillation phase.

3 Data

3.1 Distillation data

For distillation, we created the transfer sets by mixing two types of data with different distributions:

- **Generic data:** This data set consisted of Wikipedia and Common Crawl processed by an in-house tokenizer.
- **Task-specific data:** This in-house data set comprised of de-identified utterances from a voice assistant across domains of interest. The text data collected here was the output of an Automatic Speech Recognition (ASR) model, which assigned a confidence score per utterance. In order to retain only the highest quality data, we filtered it by an ASR score threshold. The data was de-identified, prior to use.

Our distilled students were trained as part of a larger program resulting in a collection of nine European and Indic languages being used for distillation. The language list and counts are shown in [Table A1](#).

We built transfer sets that had three ratios of generic to task-specific data: (1) generic-only (baseline) (2) 7:3 generic to task-specific, to mimic the commonly encountered low task-specific data setting and (3) task-specific-only. To have a comparable distribution of data from each language, we created samples of equal size for each language using either generic only, task-specific only or combining both the generic and the task-specific data based on the targeted ratio. Upsampling is used when a source data set contains a number less than the required number. The 7:3 ratio consisted of Wikipedia, Common Crawl and task-specific data upsampled to counts of 35M, 35M and 30M respectively, for each language. For two languages Indian-English and Marathi, where some data constituents were unobtainable, available data was used in proportion (see [Table A1](#)). Once the data sets were created with the targeted mixing ratio, they were split into train and validation sets with a ratio of 0.995:0.005 and then used in the transfer sets.

3.2 Data for downstream tasks

We evaluated our multilingual distilled students in the context of two commonly utilized NLU tasks of interest, DC and ICNER. We limit the scope of our evaluation to just four languages German, French, Italian and Spanish. Our finetuning data consisted of 26 domains (see fractional utterance counts in Table A4) across each language, with each domain comprising a set of intents (similar to Su et al. 2018). As with the task-specific data used in our transfer sets, this data has also been de-identified prior to use.

It is important to note that, although collected over non-overlapping time intervals (and thus consisting of different absolute counts), the finetuning data was from the same distribution as the task-specific data described in Section 3.1. We sampled the finetuning data so as to have equal counts across each domain in all four languages (see Appendix A.1 for the evaluation data set sampling strategy). We then combined all languages and split the data into proportions of 80:10:10 for train, validation and test, respectively.

For the DC task, we classified the input utterances into one of the 26 domains. Therefore, the DC model is trained using the combined training data from the four languages across all domains and is tested on language-specific test data sets. For the joint ICNER task, we classified each utterance within a domain to its corresponding intent and also recognized its named-entities. For this task, we trained a model per domain, using the combined training data from the four languages for that domain. The model was evaluated using language-specific test data sets for that domain. We present results on two types of test sets. *test* comprises of the full test set obtained from the split above while *tail_test* is the subset of data points within *test* that have a frequency of occurrence less than or equal to 3. The relative data proportions used can be found in the Appendix (Table A4).

4 Models

Figure 1 shows a schematic of the models and experimental setup described in this section.

4.1 Distilled students and baselines

We use a 170 million parameter teacher (170M-teacher) that was prepared using Wikipedia, Common Crawl and mC4 (Xue et al., 2021) data. See Appendix A.2 for details on teacher prepara-

tion. From this teacher, we distilled a total of five students. We use our three transfer sets described in Section 3.1, i.e. (1) generic-only (2) 7:3 (generic:task-specific) and (3) task-specific-only, to distill the first three students. We refer to the student distilled using (1) as the generic-distilled baseline. The latter two are referred to as experiment-7:3 and experiment-task-specific-only; the naming aligned with the transfer set used. In addition to these, we create another two students where the teacher was finetuned using an MLM task before being used for distillation. In each case, the teacher was finetuned for 15625 steps using the same transfer set that was used for the subsequent distillation. We refer to these two students as experiment-7:3-FT and experiment-task-specific-only-FT. The teacher finetuning was run on a p3.16X instance with an average run time of approximately 45 hours. We collectively refer to all distilled students that are not a baseline as experimental students.

The architectures of our teacher and students are as follows. As in the paper by Devlin et al. (2019), we denote the number of layers (i.e., Transformer blocks) as L , the hidden size as H , and the number of self-attention heads as A .

- 170M-teacher: $L=16$, $H=1024$, $A=16$, feed-forward/filter size=3072, total parameters=170M
- Students: $L=4$, $H=768$, $A=16$, feed-forward/filter size=1200, total parameters=17M

For a description of the distillation setup, see Appendix A.3. Distillation was run for 1 epoch with each student extracted at 78125 steps, which equates to approximately 80M data points seen. We ran distillation on a single p3.16X instance utilizing 8 GPUs with batch-size of 2 and gradient accumulation at every 64 steps. The average run time was approximately 195 hours. Note that each distillation run used only a sample of the full data set mentioned in Section 3.1, determined by the step count. However, since the data is sampled uniformly, the language ratios and the generic:task-specific data ratio stays consistent during training.

In addition to the distilled baseline, we also created another baseline (*without distillation*) that was directly pretrained using the generic-only data. The architecture and size of this baseline was identical to that of the distilled students and it is referred to, here onward, as the directly-pretrained baseline.

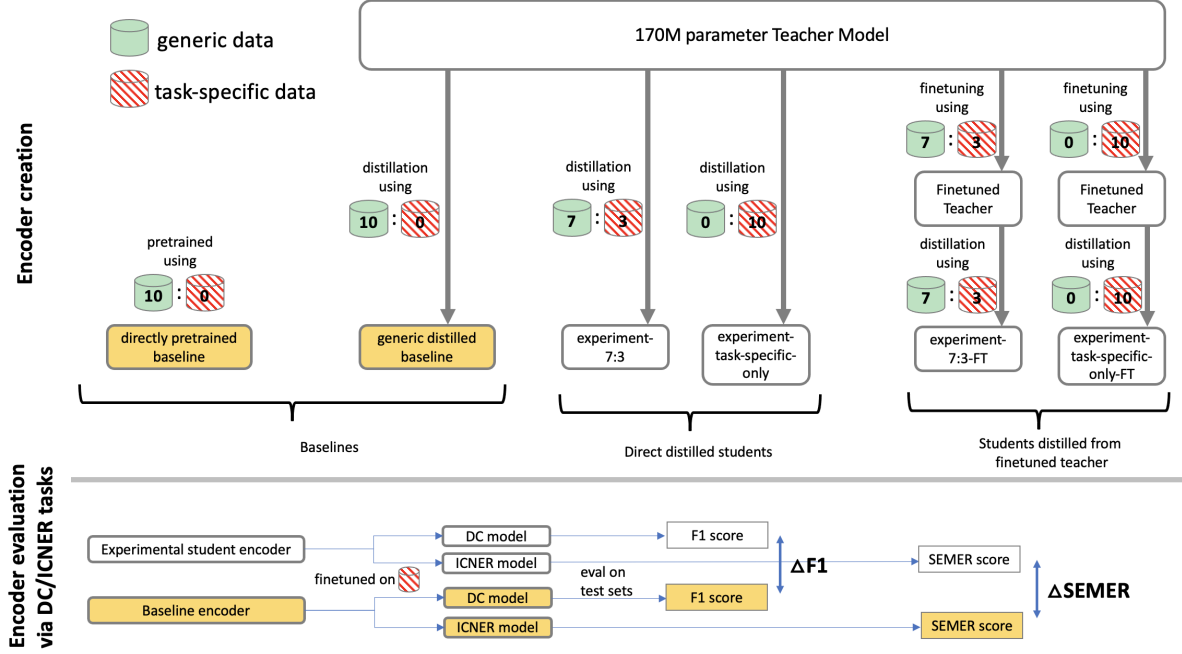


Figure 1: A schematic of the models that we present in this paper and how they are evaluated.

We used this baseline to observe performance differences between models that use students distilled from the large teacher and those that use a directly pretrained encoder.

4.2 DC and ICNER models

In order to evaluate the impact of the different transfer sets on our targeted downstream NLU tasks, we finetune the experimental students and baselines toward DC and ICNER tasks. Each DC model consisted of an encoder, embedding and positional embedding obtained from an experimental student or baseline combined with a decoder consisting of an MLP classifier for domain prediction with layer size 128, dropout set at 0.1 and ReLU activation. Each ICNER model consisted of the same encoder, embedding and positional embeddings used for the corresponding DC model with an MLP classifier output layer for the IC task with layer size 128, dropout set at 0.1 and ReLU activation and a CRF sequence-labeler output layer for the NER task with layer size 256, dropout set at 0.1 and GeLU activation. We trained each DC model for 1 epoch and each ICNER model for 4 epochs.

Evaluation: The DC performance was evaluated using the F1 score while the ICNER performance was evaluated using the Semantic Error Rate (SemER; Su et al., 2018, Varada et al., 2020, Peris et al., 2020). The definition of SemER is

$$SemER = \frac{(D + I + S)}{(C + D + S)} \quad (1)$$

where D (deletion), I (insertion), S (substitution), C (correct slots). The Intent was treated as a slot in this metric, and the Intent error was considered as a substitution.

5 Experiments

5.1 Experimental results

In this section, note that *model* refers a model that uses an experimental student or baseline encoder and has been finetuned towards a DC or ICNER task. Experimental models comprise of experimental student encoders and baseline models comprise of baseline encoders (see lower panel in Figure 1).

We used data across 26 domains to train and evaluate the DC and ICNER models (see Section 3.2). We compare the performance of each experimental model against the two baseline models (see Section 4.1). The improvements we quote in this section are $\Delta F1$ (\uparrow) (*higher is better*) and $\Delta SemER$ (\downarrow) (*lower is better*; we use the weighted average of SemER across all domains) for DC and ICNER respectively, measured against the baseline models (Tables 1, 2, A2, A3).

The results in Tables 1 and 2 show that in general for both DC and ICNER tasks, all experimental students distilled with a mix of task-specific data

Distilled encoder	Baseline	Test Set	German (%)	French (%)	Italian (%)	Spanish (%)
experiment-7:3	generic distilled	test	0.19 ± 0.02	0.19 ± 0.04	0.21 ± 0.03	0.24 ± 0.03
experiment-task-specific-only	generic distilled	test	0.51 ± 0.01	0.54 ± 0.03	0.47 ± 0.03	0.55 ± 0.03
experiment-7:3-FT	generic distilled	test	0.31 ± 0.03	0.3 ± 0.02	0.31 ± 0.03	0.35 ± 0.03
experiment-task-specific-only-FT	generic distilled	test	0.69 ± 0.02	0.79 ± 0.03	0.7 ± 0.03	0.79 ± 0.02
experiment-7:3	generic distilled	tail test	0.34 ± 0.05	0.31 ± 0.09	0.42 ± 0.05	0.42 ± 0.05
experiment-task-specific-only	generic distilled	tail test	1.0 ± 0.03	1.05 ± 0.04	1.02 ± 0.06	1.07 ± 0.06
experiment-7:3-FT	generic distilled	tail test	0.56 ± 0.06	0.61 ± 0.04	0.65 ± 0.06	0.6 ± 0.07
experiment-task-specific-only-FT	generic distilled	tail test	1.38 ± 0.05	1.51 ± 0.06	1.48 ± 0.06	1.51 ± 0.05

Table 1: Relative DC $\Delta F1$ (\uparrow), measured against the generic distilled baseline for each experimental student (positive is better). We run three iterations of each experimental student and show the percentage change of their means and its standard deviation.

Distilled encoder	Baseline	Test Set	German (%)	French (%)	Italian (%)	Spanish (%)
experiment-7:3	generic distilled	test	-0.55 ± 0.09	-0.31 ± 0.07	-0.17 ± 0.12	-0.17 ± 0.09
experiment-task-specific-only	generic distilled	test	-1.25 ± 0.07	-0.81 ± 0.07	-0.56 ± 0.08	-1.3 ± 0.09
experiment-7:3-FT	generic distilled	test	-0.83 ± 0.09	-0.49 ± 0.13	-0.06 ± 0.14	-0.58 ± 0.09
experiment-task-specific-only-FT	generic distilled	test	-1.57 ± 0.15	-1.18 ± 0.07	-0.6 ± 0.25	-1.26 ± 0.04
experiment-7:3	generic distilled	tail test	-0.49 ± 0.07	-0.31 ± 0.06	-0.16 ± 0.09	-0.23 ± 0.11
experiment-task-specific-only	generic distilled	tail test	-1.19 ± 0.05	-0.86 ± 0.07	-0.67 ± 0.08	-1.44 ± 0.09
experiment-7:3-FT	generic distilled	tail test	-0.83 ± 0.09	-0.52 ± 0.1	-0.13 ± 0.09	-0.65 ± 0.11
experiment-task-specific-only-FT	generic distilled	tail test	-1.53 ± 0.12	-1.26 ± 0.07	-0.79 ± 0.14	-1.32 ± 0.06

Table 2: Relative ICNER ΔSemER (\downarrow), measured against the generic distilled baseline for each experimental student (negative is better). As with DC, we run three iterations of the experimental students and show the percentage change of their means and its standard deviation. In calculating these percentage changes, we use the weighted average of the SemER for each domain in a given language, as the overall SemER in that language.

(30% or 100%) perform significantly better than the generic distilled baseline. We further observe that models with encoders distilled with task-specific-only data yields the best overall performance which means that, in our setup, students learn better using target-domain data even if it comes at the price of noisier teacher predictions.

For all four languages across DC and for three out of four languages across ICNER, the best performances are observed with student models that were distilled from the *finetuned* teacher. This confirms that the additional step of finetuning the teacher and adapting it to the task-specific dataset, results in students that perform better on the intended downstream tasks.

We also note that across all task, language and test set combinations, the improvements seen against the directly pretrained baseline (see Tables A2 and A3) are larger than the improvements seen against the generic distilled baseline. For our setup, this shows that distilling from a large LM can benefit downstream tasks as opposed to using a similar-sized encoder pretrained from scratch; in other words our findings suggest that it is *better to distill* than to directly pretrain. However, we note

that additional resources (in our case approximately 45 p3.16X hours) are required for this.

The *tail_test*, comprising of low frequency utterances within test, provides insights on the ability of the model to generalize to rarely seen utterances. For DC, we note that the improvements on *tail_test* are significantly larger ($\sim 2X$) than the improvements seen on the test set. This indicates that prediction on examples that appear infrequently in the task-specific data benefits more, from task-specific data being included in the distillation process.

5.2 Dataset similarity and its correlation to SemER improvements for ICNER

To further explore our conclusion that *students learn better using target-domain data* we explore how ΔSemER for each domain, correlate to the similarity of the domain’s data to the generic data. Note that, here, negative ΔSemER represents improvements of the experimental students against the generic distilled baseline while the opposite is true for positive ΔSemER . SemER results are from the *test* set.

The hypothesis here is that the more distant a domain is from the generic data, the more value we should see in adding this domain’s data to the distil-

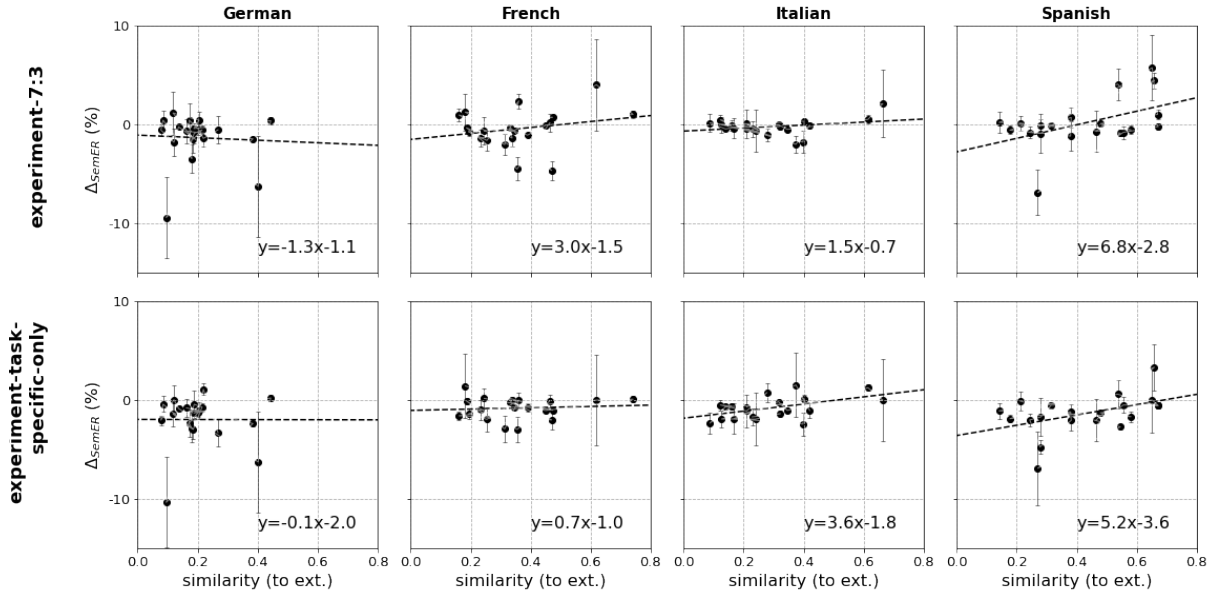


Figure 2: Cosine similarity of *tf-idf* vectors vs. change in SemER for each domain for languages German, French, Italian and Spanish. We represent only domains with >1000 test utterances to avoid noise added by smaller domains which have higher variability.

lation transfer set, even though teacher predictions might be noisy. We note here that we calculate cosine similarity on a very rudimentary corpus-level embedding (i.e. *tf-idf*) for measuring similarity, as explained below. We leave more sophisticated similarity measurements for later work.

To calculate similarity between domain-level and generic data, we use the following process. For each domain in each of the four languages, we sample up to 100K utterances. All available data is considered for domains with <100K utterances. We then sample 50K utterances each, from the Wikipedia and Common Crawl data sets of the corresponding language. We create a *tf-idf* vector for each sampled dataset and calculate their cosine similarity as a measure of dataset similarity. In order to account for any variability associated with the sampling, we repeat the process 3 times and obtain the mean similarity and the standard deviation per domain. We plot dataset similarity against ΔSemER (a single point represents one domain and a panel represents a language as seen in Figure 2). We neglect domains with lower data and thus high variability and fit a line to show how ΔSemER correlates to dataset similarity.

In Figure 2, we observe that a majority of cases (all except German) show a positive correlation. A positive correlation shows that domains that are *less* similar to Wikipedia/CommonCrawl have relatively larger improvement in SemER, when

compared to domains that are more similar to Wikipedia/CommonCrawl. This suggests that the addition of task-specific data in the distillation transfer sets helps domains that are less similar to the generic data available for distillation, even though teacher predictions on them will be more noisy.

It should be noted that the domains of the one exception, German, display low similarity values across the board unlike the other languages which show a wider spread (German has 65% of domains < 0.2 whereas French, Italian and Spanish has 23%, 31% and 12% < 0.2 respectively). The lack of domains with high similarity might explain the failure for a stable correlation to be observed in German.

6 Conclusions

We have explored how the use of transfer sets that comprise different ratios of generic to task-specific data, impacts downstream results. Encoders distilled from a large teacher perform better than ones trained from scratch, showing that it is *better to distill* than to directly pretrain, when possible. The largest benefits are shown when using the downstream task’s unlabelled data to distill, a student despite noisier teacher predictions. We also find that domains with data that are dissimilar to the generic data show greater performance improvements against a generic baseline when using a stu-

dent distilled using task-specific data. These improvements further confirm that distilling using target-domain data can be helpful for downstream performance. Finally, we show that if costs permit, teacher-adaptation to the target-domain data via finetuning can result in improved student performance across downstream tasks.

Acknowledgements

We thank Karolina Owczarzak, Fabian Triefenbach and Rahul Gupta for their support of this work and Wael Hamza for helpful discussions on the topics covered.

References

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. 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.
- Jack G. M. FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, David Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, Rakesh Chada, Amit Chauhan, Luoxin Chen, Anurag Dwarakanath, Satyam Dwivedi, Turan Gojayev, Karthik Gopalakrishnan, Thomas Gueudré, Dilek Z. Hakkani-Tür, Wael Hamza, Jonathan Hueser, Kevin Martin Jose, Haidar Khan, Bei Liu, Jianhua Lu, A. Manzotti, Pradeep Natarajan, Karolina Owczarzak, Gokmen Oz, Enrico Palumbo, Charith S. Peris, Chandan Prakash, Stephen Rawls, Andrew Rosenbaum, Anjali Shenoy, Saleh Soltan, Mukund Sridhar, Lizhen Tan, Fabian Triefenbach, Pang Wei, Haiyang Yu, Shuai Zheng, Gokhan Tur, and Premkumar Natarajan. 2022. [Alexa teacher model: Pretraining and distilling multi-billion-parameter encoders for natural language understanding systems](#). *ArXiv*, abs/2206.07808.
- Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. [Compressing deep convolutional networks using vector quantization](#).
- 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.
- Song Han, Jeff Pool, John Tran, and William J. Dally. 2015. [Learning both weights and connections for efficient neural networks](#).
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. [Distilling the knowledge in a neural network](#).
- 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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.
- Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2019. [Improved knowledge distillation via teacher assistant](#).
- Charith Peris, Gokmen Oz, Khadige Abboud, Venkata sai Varada Varada, Prashan Wanigasekara, and Haidar Khan. 2020. [Using multiple ASR hypotheses to boost i18n NLU performance](#). In *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*, pages 30–39, Indian Institute of Technology Patna, Patna, India. NLP Association of India (NLP AI).
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#).
- 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.
- Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. [Fitnets: Hints for thin deep nets](#).
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).
- Saleh Soltan, Haidar Khan, and Wael Hamza. 2021. [Limitations of knowledge distillation for zero-shot transfer learning](#). In *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*, pages 22–31, Virtual. Association for Computational Linguistics.

Chengwei Su, Rahul Gupta, Shankar Ananthkrishnan, and Spyridon Matsoukas. 2018. A re-ranker scheme for integrating large scale nlu models. *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 670–676.

Venkat Varada, Charith Peris, Yangsook Park, and Christopher Dipersio. 2020. [Using alternate representations of text for natural language understanding](#). In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 1–10, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

A Appendix

A.1 Data for finetuning DC and ICNER models

For finetuning our distilled students for DC and ICNER, we use labelled datasets from four languages (German, French, Italian and Spanish) each consisting the same 26 domains (Table A4) and each domain supporting a set of intents (similar to Su et al. 2018). In order to have equivalent utterance counts across domains for each language, we used a stratified sampling strategy as follows. First, we ranked each language per domain based on its utterance counts. In order to prevent heavy upsampling or downsampling in any single language when creating equivalently sampled domains, we picked the language that had the second highest utterance counts in most domains (in our case French). We sampled utterances from the domains of other languages to match the domain-level utterance frequency distribution of French (i.e. random sample utterances with replacement, from each domain in each language until that number matches the utterance count of the respective domain in French). We then combined all languages and split the data into proportions of 80:10:10 for train, validation and test, respectively.

A.2 Teacher model

The 170M-teacher used in this work was, itself, a student that was distilled from a larger model with 2 billion parameters (see Stage 1 pretraining section in FitzGerald et al. (2022) for details on creation

and architecture). The 170M-teacher was distilled using a transfer set that comprised Wikipedia, Common Crawl and mC4 (Xue et al., 2021) data. Picking this intermediate-sized model helped us avoid potential performance degradation due to having too large a size gap between teacher and student (Mirzadeh et al., 2019).

A.3 Student setup

For our single-stage distillation setup, we skip the generic distillation phase done by Jiao et al. (2020) and use a non-finetuned teacher model to directly distill our students. In addition, as a sanity check, we also explore distillation from a finetuned teacher model to verify improved student performance across tasks. Similar to the hidden states based distillation followed in TinyBERT (Jiao et al., 2020), we mapped the student layers [0, 1, 2, 3] to learn from the teachers hidden layers [3, 7, 11, 15], respectively. We ignored attention based distillation (Jiao et al., 2020) since we did not observe significant improvements by using it. We also penalized the soft cross-entropy loss between the student network’s logits against the teacher’s logits, to fit the students predictions to those of the teacher as in Hinton et al. (2015). We use a MLM objective for the distillation process. In our loss, we weight the hidden layer matching, the logit matching and the MLM at a 1:2:1 ratio.

	Common Crawl (cc100)	Wikipedia	Task Specific Data
German	12,045,483	2,731,840	32,081,929
French	13,323,804	2,174,531	21,278,820
Italian	7,131,950	1,278,255	31,013,233
Spanish	11,690,123	1,825,389	22,054,722
English	14,330,660	6,360,372	19,576,081
English (IN)	-	-	27,406,082
Hindi	2,538,698	94,891	21,315,004
Tamil	919,763	66,190	-
Tamil (MT)	-	-	18,414,285
Telugu	378,812	77,179	-
Telugu (MT)	-	-	18,895,352
Marathi	263,189	21,705	-

Table A1: Raw data counts used for transfer set creation. Counts represent the number of sentences for generic data and the number of de-identified utterances for task-specific data. For task-specific data for Telugu and Tamil, machine-translated (MT) data from Indian English was used. Only task-specific data was used for Indian English because Wikipedia and Common Crawl data were not available. Only generic data was used for Marathi as the translation system used for this work did not support the language as yet.

Distilled encoder	Baseline	Test Set	German (%)	French (%)	Italian (%)	Spanish (%)
experiment-7:3	directly pretrained	test	0.31 ± 0.02	0.19 ± 0.04	0.32 ± 0.01	0.28 ± 0.03
experiment-task-specific-only	directly pretrained	test	0.63 ± 0.01	0.54 ± 0.02	0.57 ± 0.01	0.6 ± 0.02
experiment-7:3-FT	directly pretrained	test	0.42 ± 0.03	0.3 ± 0.02	0.41 ± 0.01	0.4 ± 0.02
experiment-task-specific-only-FT	directly pretrained	test	0.81 ± 0.01	0.79 ± 0.03	0.8 ± 0.02	0.83 ± 0.01
experiment-7:3	directly pretrained	tail test	0.57 ± 0.04	0.36 ± 0.08	0.57 ± 0.01	0.45 ± 0.03
experiment-task-specific-only	directly pretrained	tail test	1.23 ± 0.02	1.1 ± 0.02	1.17 ± 0.03	1.09 ± 0.04
experiment-7:3-FT	directly pretrained	tail test	0.78 ± 0.05	0.66 ± 0.02	0.8 ± 0.03	0.62 ± 0.06
experiment-task-specific-only-FT	directly pretrained	tail test	1.61 ± 0.04	1.56 ± 0.05	1.63 ± 0.03	1.54 ± 0.04

Table A2: Relative DC $\Delta F1$ (\uparrow) measured against the directly pretrained baseline for each experimental student (positive is better)

Distilled encoder	Baseline	Test Set	German (%)	French (%)	Italian (%)	Spanish (%)
experiment-7:3	directly pretrained	test	-1.73 ± 0.06	-0.7 ± 0.04	-2.3 ± 0.11	-2.44 ± 0.09
experiment-task-specific-only	directly pretrained	test	-2.42 ± 0.02	-1.19 ± 0.04	-2.69 ± 0.07	-3.54 ± 0.08
experiment-7:3-FT	directly pretrained	test	-2.01 ± 0.06	-0.88 ± 0.12	-2.2 ± 0.13	-2.84 ± 0.09
experiment-task-specific-only-FT	directly pretrained	test	-2.74 ± 0.13	-1.57 ± 0.05	-2.73 ± 0.24	-3.5 ± 0.04
experiment-7:3	directly pretrained	tail test	-1.72 ± 0.05	-0.7 ± 0.05	-2.62 ± 0.09	-2.62 ± 0.09
experiment-task-specific-only	directly pretrained	tail test	-2.41 ± 0.02	-1.24 ± 0.05	-3.11 ± 0.07	-3.8 ± 0.07
experiment-7:3-FT	directly pretrained	tail test	-2.05 ± 0.08	-0.9 ± 0.09	-2.59 ± 0.08	-3.03 ± 0.09
experiment-task-specific-only-FT	directly pretrained	tail test	-2.74 ± 0.11	-1.64 ± 0.05	-3.23 ± 0.13	-3.68 ± 0.03

Table A3: Relative ICNER ΔSemER (\downarrow) measured against the directly pretrained baseline for each experimental student (negative is better). In calculating these percentage changes, we use the weighted average of the SemER for each domain in a given language, as the overall SemER in that language.

Table A4: Fractions of finetuning data used per domain. Note that the fraction in each cell represents the utterance count for that partition, for that domain, as a fraction of the total utterance count in that language. As mentioned in Section 3.2, these fractions are not based on the counts in Table A1

	train				validation				test				tail_test			
	German	French	Italian	Spanish	German	French	Italian	Spanish	German	French	Italian	Spanish	German	French	Italian	Spanish
	Domain 1	0.014%	0.014%	0.014%	0.014%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%	0.002%
Domain 2	0.634%	0.634%	0.630%	0.632%	0.078%	0.078%	0.078%	0.077%	0.081%	0.080%	0.080%	0.079%	0.059%	0.063%	0.058%	0.055%
Domain 3	0.663%	0.664%	0.662%	0.663%	0.082%	0.082%	0.082%	0.084%	0.083%	0.083%	0.082%	0.082%	0.059%	0.064%	0.062%	0.060%
Domain 4	0.072%	0.072%	0.072%	0.071%	0.009%	0.009%	0.009%	0.009%	0.009%	0.009%	0.009%	0.009%	0.007%	0.008%	0.007%	0.007%
Domain 5	6.024%	6.037%	6.024%	6.026%	0.760%	0.754%	0.757%	0.757%	0.755%	0.751%	0.756%	0.751%	0.346%	0.343%	0.355%	0.377%
Domain 6	0.524%	0.525%	0.525%	0.523%	0.067%	0.065%	0.065%	0.064%	0.067%	0.066%	0.065%	0.067%	0.028%	0.033%	0.036%	0.034%
Domain 7	0.369%	0.369%	0.368%	0.368%	0.045%	0.045%	0.046%	0.048%	0.047%	0.047%	0.046%	0.046%	0.031%	0.036%	0.027%	0.030%
Domain 8	1.043%	1.042%	1.041%	1.042%	0.131%	0.131%	0.132%	0.131%	0.133%	0.128%	0.132%	0.132%	0.077%	0.060%	0.066%	0.066%
Domain 9	18.474%	18.462%	18.464%	18.466%	2.307%	2.300%	2.299%	2.305%	2.305%	2.310%	2.312%	2.303%	2.02%	2.63%	3.42%	2.92%
Domain 10	0.005%	0.005%	0.005%	0.005%	0.001%	0.000%	0.001%	0.001%	0.001%	0.001%	0.001%	0.001%	0.001%	0.001%	0.001%	0.001%
Domain 11	0.462%	0.465%	0.464%	0.464%	0.057%	0.058%	0.057%	0.057%	0.057%	0.059%	0.059%	0.056%	0.029%	0.034%	0.021%	0.034%
Domain 12	9.737%	9.720%	9.735%	9.730%	1.218%	1.217%	1.217%	1.216%	1.216%	1.214%	1.209%	1.216%	0.401%	0.373%	0.352%	0.377%
Domain 13	5.915%	5.916%	5.923%	5.917%	0.741%	0.740%	0.737%	0.739%	0.736%	0.735%	0.733%	0.740%	0.511%	0.497%	0.478%	0.478%
Domain 14	1.454%	1.456%	1.457%	1.455%	0.183%	0.184%	0.183%	0.184%	0.181%	0.182%	0.183%	0.182%	0.162%	0.171%	0.172%	0.155%
Domain 15	13.658%	13.667%	13.666%	13.663%	1.703%	1.717%	1.705%	1.710%	1.709%	1.710%	1.714%	1.708%	0.998%	1.101%	1.100%	1.026%
Domain 16	3.470%	3.471%	3.472%	3.474%	0.435%	0.434%	0.437%	0.432%	0.435%	0.435%	0.435%	0.439%	0.209%	0.245%	0.231%	0.234%
Domain 17	0.028%	0.029%	0.029%	0.029%	0.004%	0.004%	0.004%	0.003%	0.004%	0.003%	0.004%	0.004%	0.003%	0.003%	0.003%	0.003%
Domain 18	0.655%	0.656%	0.657%	0.659%	0.082%	0.083%	0.083%	0.083%	0.082%	0.082%	0.084%	0.081%	0.026%	0.029%	0.030%	0.021%
Domain 19	0.428%	0.429%	0.428%	0.427%	0.053%	0.053%	0.053%	0.053%	0.055%	0.053%	0.053%	0.052%	0.046%	0.046%	0.046%	0.047%
Domain 20	10.209%	10.210%	10.211%	10.214%	1.271%	1.276%	1.283%	1.277%	1.273%	1.277%	1.272%	1.279%	0.678%	0.642%	0.706%	0.923%
Domain 21	0.046%	0.045%	0.045%	0.044%	0.006%	0.006%	0.005%	0.006%	0.006%	0.006%	0.006%	0.006%	0.003%	0.003%	0.003%	0.002%
Domain 22	1.247%	1.253%	1.250%	1.252%	0.159%	0.159%	0.159%	0.156%	0.156%	0.160%	0.157%	0.158%	0.128%	0.129%	0.107%	0.120%
Domain 23	0.223%	0.223%	0.222%	0.221%	0.028%	0.029%	0.029%	0.028%	0.028%	0.028%	0.029%	0.028%	0.026%	0.026%	0.026%	0.025%
Domain 24	2.986%	2.981%	2.978%	2.984%	0.372%	0.369%	0.372%	0.370%	0.373%	0.373%	0.372%	0.375%	0.265%	0.289%	0.289%	0.270%
Domain 25	1.617%	1.614%	1.617%	1.613%	0.202%	0.203%	0.204%	0.203%	0.202%	0.202%	0.201%	0.202%	0.083%	0.091%	0.098%	0.083%
Domain 26	0.042%	0.042%	0.042%	0.042%	0.006%	0.005%	0.005%	0.005%	0.005%	0.005%	0.006%	0.005%	0.004%	0.003%	0.004%	0.004%
Total Fraction	80.000%	80.000%	80.000%	80.000%	10.000%	10.000%	10.000%	10.000%	10.000%	10.000%	10.000%	10.000%	4.386%	4.554%	4.622%	4.708%

Exploiting In-Domain Bilingual Corpora for Zero-Shot Transfer Learning in NLU of Intra-Sentential Code-Switching Chatbot Interactions

Maia Aguirre, Manex Serras, Laura García-Sardiña, Jacobo López-Fernández
Ariane Méndez and Arantza del Pozo

Vicomtech Foundation, Basque Research and Technology Alliance (BRTA)

Parque Científico y Tecnológico de Gipuzkoa, Paseo Mikeletegi 57, Donostia / San Sebastián (Spain)
{magirre, mserras, lgarcias, jlopez, amendez, adelpozo}@vicomtech.org

Abstract

Code-switching (CS) is a very common phenomenon in regions with various co-existing languages. Since CS is such a frequent habit in informal communications, both spoken and written, it also arises naturally in Human-Machine Interactions. Therefore, in order for natural language understanding (NLU) not to be degraded, CS must be taken into account when developing chatbots. The co-existence of multiple languages in a single NLU model has become feasible with multilingual language representation models such as mBERT. In this paper, the efficacy of zero-shot cross-lingual transfer learning with mBERT for NLU is evaluated on a Basque-Spanish CS chatbot corpus, comparing the performance of NLU models trained using in-domain chatbot utterances in Basque and/or Spanish without CS. The results obtained indicate that training joint multi-intent classification and entity recognition models on both languages simultaneously achieves best performance, better capturing the CS patterns.

1 Introduction

Multilingual speakers outnumber monolingual speakers in the world (Tucker, 2001). In regions with various coexisting languages, a common feature of natural interactions amongst speakers is the continuous casual switching between the concerned languages or “codes”. This phenomenon, known as Code-Switching (CS), is very frequent in both spoken and written informal interactions (Ahn et al., 2020). In fact, the percentage of CS in social networks ranges from 14.5% to 49.06% for the corpora explored in Gambäck and Das (2016). Moreover, in the study Al-Qaysi and Al-Emran that explores the educators and learners’ attitudes towards using CS online, 86.40% of the students and 81% of the teachers claim to actively code-switch while chatting on social networks.

As is to be expected, CS also arises spontaneously during human-machine interactions such

as conversing with chatbots (i.e., conversational agents). This being the case, Bawa et al. (2020) reveal that these interlocutors strongly prefer chatbots that do understand CS.

The most common strategy employed by chatbots to understand the interlocutor is to employ intent and entity-based annotation schemata. This involves intent –communicative purpose– detection and entity –key words– classification processes (Tur et al., 2010). For example, in the utterance “I want to have an Italian meal that does not exceed 15 euros”, the *intent* behind the user’s query is to *order food* given the *entity* labels *cheap* and *italian*.

Allowing multilingual interactions with chatbots involves running a language identifier prior to each speaker turn and executing language-specific Natural Language Understanding (NLU) models. However, this approach is only effective for intersentential CS, where the code alternation happens at utterance boundaries. In the case of intrasentential CS, where the same utterance contains words or phrases belonging to two or more languages (Gumperz, 1982), detecting every intent and entity of the utterance poses a major challenge for existing algorithms (Banerjee et al., 2018). The previous sentence with intra-sentential CS would be “*Quiero an Italian meal que no supere los 15 euros*” (Spanish in italics). In this case, neither an NLU model in English nor an NLU model in Spanish would detect that the intent is *order food*.

One strategy to solve this problem would be treating CS as a language itself and training an NLU model with examples containing CS. However, collecting data with CS is quite challenging, as it hardly exists in written form and requires bilingual annotators. Instead, obtaining labeled monolingual data in multiple languages is easier and zero-shot cross-lingual transfer learning (TL) has been proved to perform well across different Natural Language Processing (NLP) tasks, including NLU (M’hamdi et al., 2021). Thus, this TL

approach would be more practical to exploit in real-world industrial chatbot applications to be deployed in CS regions. Unfortunately, such methodology has not yet been explored in the literature, mainly due to the scarcity of real intra-sentential CS datasets.

In this paper, a Basque-Spanish CS corpus is exploited to evaluate different zero-shot cross-lingual TL experiments aiming to examine whether such multilingual training methodologies are capable of addressing the NLU problem of intra-sentential CS chatbot interactions. For this purpose, the effectiveness of three multilingual models is analysed in their ability to understand CS: one fine-tuned using a chatbot corpus in Basque, another one fine-tuned on a corpus of Spanish chatbot utterances and a third one fine-tuned using both corpora simultaneously. It is important to underline that none of the models was exposed to CS during training, and that the monolingual Basque and Spanish training corpora belong to the same domain of the CS corpus used for testing purposes. Through this comparison it has been determined that models fine-tuned on in-domain bilingual corpora simultaneously are able to generate cross-lingual bonds and perform better against CS. The fact that zero-shot fine-tuning of multilingual BERT models on both monolingual languages enhances their effectiveness in understanding CS stands as one of the main contributions of this work.

The remaining of the paper is structured as follows: Section 2 reviews recent work in the area of joint intent and entity detection and multilingual models. Section 3 analyzes the main characteristics of the corpora used both for training and evaluation purposes. Section 4 presents the architecture and specifications of the joint intent and entity detection implementation employed. Section 5 shows the results obtained in the different experiments carried out and, finally, Section 6 highlights the main conclusions and proposes tentative lines for future work.

2 Related Work

Intent Detection and Entity Classification

The traditional way of approaching intent detection and entity classification tasks for NLU is to address them separately. However, treating each task as an individual problem leads to inefficient usage of training resources. Among others, [Chen et al. \(2019\)](#); [Lorenc \(2021\)](#) have shown that com-

binning intent and entity recognition in a single system achieves significant improvements in both tasks with lower computational resources. [Cai et al. \(2022\)](#) and [Qin et al. \(2020\)](#) propose novel methods that consider joint learning of both tasks by correlating the intents and entities and reach new state-of-the-art performance. In addition, [Castellucci et al. \(2019\)](#) have explored how these joint approaches also perform better in multilingual settings.

Multilingual models

Contextualised multilingual models, such as mBERT and XLM-R ([Conneau and Lample, 2019](#)), have achieved state-of-the-art results in monolingual and multilingual tasks on NLU benchmark tests ([Wang et al., 2019](#); [Hu et al., 2020](#); [Liu et al., 2020](#)). However, the effectiveness of NLU models on CS interactions remains unknown ([Winata et al., 2021](#)).

Still, there have been several attempts to use multilingual representations to encode CS sentences ([Srinivasan, 2020](#); [Aguilar et al., 2020](#); [Khanuja et al., 2020](#)), showing promising results and surpassing previously achieved performances ([Aguilar et al., 2020](#); [Khanuja et al., 2020](#)).

Recent work has shown that, even if the embeddings across the 12 multi-head attention layers of mBERT are clustered across languages ([Krishnan et al., 2021](#)), they can be split into two components: a language-specific one and a language-neutral one ([Krishnan et al., 2021](#); [Libovický et al., 2020](#); [Tanti et al., 2021](#)). [Pires et al. \(2019\)](#) have also found that a shared subspace representing relevant linguistic information is common to cross-lingual BERT representations. Likewise, [Chi et al. \(2020\)](#) claim that part of the representation space of the syntactic level of mBERT is shared between languages and identify that mBERT has a cross-linguistic clustering of grammatical relations. In addition, [Cao et al. \(2019\)](#) suggest that mBERT also aligns semantics across languages. [Libovický et al. \(2020\)](#) use a set of semantic-oriented tasks to show that unsupervised multilingual contextual embeddings based on BERT capture similar semantic phenomena in very similar ways across languages.

Moreover, [Krishnan et al. \(2021\)](#) and [Tanti et al. \(2021\)](#) have also demonstrated that the cross-lingual capacity of mBERT models increases after fine-tuning as the models switch their ability to cluster embeddings by language to cluster them

according to the needs of the task. For example, regarding the intent detection task, the embeddings will be grouped by intents after fine-tuning.

On the other hand, several benchmarking experiments in NLP tasks other than NLU against CS test sets have shown that mBERT fine-tuning achieves the best performance compared to alternative multilingual models. [Khanuja et al. \(2020\)](#) presented the first model evaluation benchmark against CS. After testing various embedding techniques for all tasks and datasets, they concluded that the multilingual BERT model performs the best. They also demonstrate that, for most datasets, a modified version of mBERT that has been subsequently fine-tuned with synthetically generated CS data performs consistently better. [Aguilar et al. \(2020\)](#) propose another benchmark metric that combines ten corpora covering four different CS language pairs and four NLP tasks for the evaluation of linguistic CS. Superior performance of the mBERT models for each available language pair is observed across the vast majority of the tasks.

3 Data

3.1 Training and validation corpus

Three corpora have been used to train and validate the models:

1. A Basque corpus, consisting of utterances in the Basque language
2. A Spanish corpus, formed by utterances in Spanish language
3. A Bilingual corpus, grouping both corpora together

The Basque and Spanish corpora comprise a collection of text samples used to train the NLU modules of four bilingual (Basque-Spanish) chatbots. These chatbots were designed to answer specific questions related to the fields of administration, taxation, and transport. In addition, they were able to respond to greetings, requests for help, and some common social questions such as "Are you a robot?", etc. Besides their domain label, the examples in each corpus were annotated with semantic information regarding their intents and entity values. The preprocessing of the training data involved removing stress marks, capitalisation and punctuation marks, considering that users tend to write without respecting spelling rules while chatting.

The three training corpora are divided into a training set and a validation set, with a ratio of 75/25 and with an even distribution of intents and entities. The total number of unique entities is 39 and the number of unique intents is 90.

The partition size of each corpus in the training and validation sets is reflected in Table 1.

	Training	Validation
Basque corpus	1662	555
Spanish corpus	1452	485
Bilingual corpus	3114	1040

Table 1: Number of utterances associated with the training and validation set of each corpus.

3.2 Test corpus

In our experiments, The BaSCo –Basque-Spanish Code-Switching– corpus ([Aguirre et al., 2022](#)) is used to evaluate the robustness of the different NLU models against Basque-Spanish CS.

It is a compendium of 1377 utterances containing Basque-Spanish intra-sentential CS that belong to the same domain as the corpora used for training (i.e. chatbot interactions related to the fields of administration, taxation and transport) and also share the same set of intent and entity labels.

4 Implementation Strategies

The joint intent detection and entity classification NLU model developed in this work takes the one presented by [Chen et al. \(2019\)](#) and its corresponding implementation¹ as a baseline. For our experiments, the baseline model architecture has been adapted as shown in Figure 1 to support the detection of multiple intents in the same utterance.

This adaptation has involved two major changes. On the one hand, intent label representation has been adapted to allow assigning more than one intent to each utterance. For this purpose, *one-hot*² encoding has been adopted and consequently the activation function of the final layer has been changed from softmax to sigmoid. On the other hand, the loss function selected for multi-intent classification optimisation has been changed to *binary cross-entropy*, as it allows each utterance to have more than one associated intent ([Ho and Wookey, 2019](#)).

¹<https://github.com/90217/joint-intent-classification-and-slot-filling-based-on-BERT>

²<https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MultiLabelBinarizer.html>

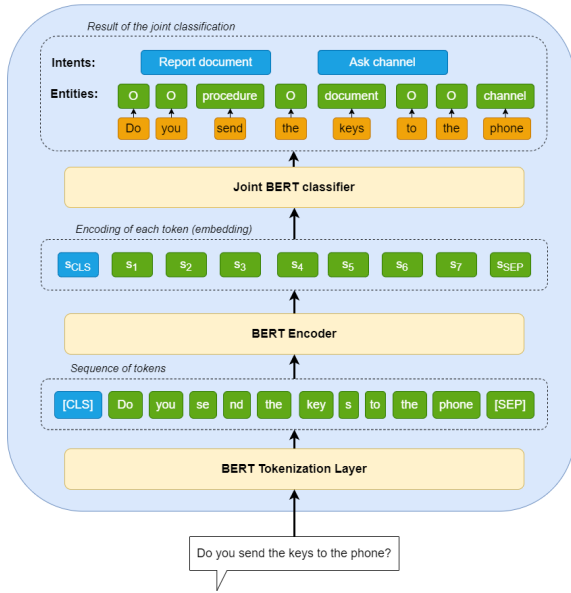


Figure 1: Architecture of the joint multi-intent and entity detection model. It consists of a first module that tokenises the input sentence and translates the information to the BERT model, which returns an embedding for each element of the sequence and a global embedding of the utterance. Entities are predicted using the individual embeddings of the sequence and intents are predicted using sentence embeddings.

This loss function predicts whether each possible intent appears in the utterance regardless of the rest of the intents. The loss function used to classify entities remains *sparse categorical cross-entropy*, since it meets the assumption that each token belongs to a single entity category. The goal of the implemented joint multi-intent and entity classifier is to minimise the sum of the two individual loss functions.

The BERT model employed is BERT-Base-Multilingual-Cased or mBERT, which has 12 layers, 768 hidden states and 12 heads. And the best performing hyperparameters are a maximum length and batch size of 128; the Adam optimiser (Kingma and Ba, 2015); a learning-rate of $9e-5$; a dropout probability of 0.1; and 50 epochs.

5 Results

5.1 Validation results

Table 2 collects the results returned by the different models when evaluated over the Basque and Spanish validation sets. This table allows to directly compare the performance of the model trained on the bilingual corpus versus the performances of the models trained on the monolingual corpora.

The model trained with the bilingual corpus uses the same train/validation partition as the models trained on the monolingual corpora (i.e., the utterances used for training and evaluating the model are identical). Still, the results obtained with the model trained on the bilingual corpus outperform the results of the models trained on the monolingual corpora as more intents are properly classified. Therefore, it can be determined that, as expected, cross-lingual learning actually happens in the model trained with both languages.

This learning improvement occurs because in the fine-tuning process the model is trained with the same set of intent and entity labels for the Basque and Spanish corpora. In this way, it learns to relate and project text entries in different languages onto a common label space.

5.2 Test results

To better assess the cross-lingual learning capabilities of the trained models, the BaSCo corpus of intrasentential Basque-Spanish CS utterances is used as test set.

The results obtained for each model are shown in Table 3. As it can be seen, the cross-lingual comprehension acquired by the model trained on the Bilingual corpus is also evidenced against the CS test set, showing a clear improvement over the models trained on the monolingual corpora with results such as:

- +38 and +17 points in F1 micro and macro metrics respectively for intent classification.
- +2 and +15 points in F1 micro and macro metrics respectively for entity classification.

A more intuitive way of visualising the cross-lingual learning of the model trained on the Bilingual corpus is by means of a two-dimensional representation of the embeddings (i.e. the result given by mBERT's *pooling layer* for each sentence input to the model). For this purpose, the t-SNE method (t-distributed Stochastic Neighbor Embedding) is used to assign each high-dimensional data vector a position in a two-dimensional map (Van der Maaten and Hinton, 2008). In this way, Figure 2 shows the two-dimensional representation that each of the three models assigns to each of the following sets: the validation set of the Basque corpus, the validation set of the Spanish corpus and the BaSCo intra-sentential CS test set.

Metrics	Validation set	Models		
		Basque	Spanish	Bilingual
Intent classifier loss	Basque corpus	0.0292	/	0.0241
	Spanish corpus	/	0.0274	0.0214

Table 2: Loss metrics of the three multilingual models: (i) fine-tuned on the Basque corpus, (ii) fine-tuned on the Spanish corpus and (iii) fine-tuned on the Bilingual corpus, in their ability to understand multiple intents and entities. The results obtained at the end of training (epoch 50) are shown for the Basque and Spanish corpus validation sets separately.

Metrics		Model		
		Basque	Spanish	Bilingual
Multiple Intent Classification	Intent classifier loss	0.0594	0.0676	0.0351
	F1 Score micro	0.4339	0.3958	0.8148
	F1 Score macro	0.2889	0.3081	0.4714
Entity Classification	F1 Score micro	0.7162	0.6567	0.7358
	F1 Score macro	0.3593	0.4420	0.5933

Table 3: Loss and F1 metrics of the three multilingual models: (i) fine-tuned on the Basque corpus, (ii) fine-tuned on the Spanish corpus, and (iii) fine-tuned on the Bilingual corpus, in their ability to understand multiple intents and entities. The results obtained at the end of training (epoch 50) are shown when evaluated over the BaSCo test set.

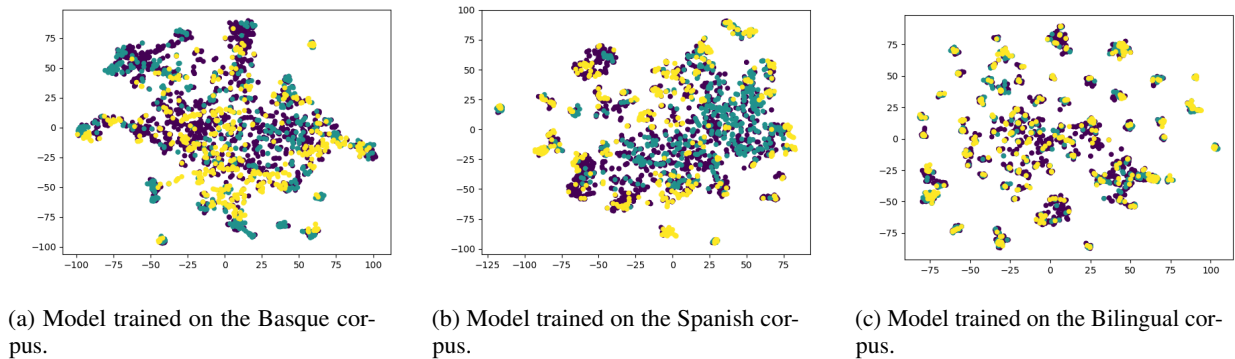


Figure 2: t-SNE representation of the sentence embeddings of the validation set of the Basque corpus (green), the validation set of the Spanish corpus (yellow), and the BaSCo test set (purple) of the models trained on the Basque, Spanish, and Bilingual corpus.

As it can be appreciated, the models trained on the monolingual Basque and Spanish corpora (Figures 2a and 2b) present a major dispersion of the points in the plane, having greater difficulty in determining clear groupings. In contrast, in Figure 2c there is a very noticeable overlapping of dots of different colours (i.e. of different languages) yielding clearly delimited groupings. These clusters have a clear semantic charge because they concentrate sentences that share the same intent. This property can be easily ascertained in the interactive web version of the figure³, where sentences with, for example,

³<https://clusters-mbert.dialogue.vicomtech.org>

the intent label “greeting” are grouped around the same point on the map, regardless of whether they are in Spanish, Basque or Basque-Spanish CS.

These results are very positive, as they show that state-of-the-art NLU models have the ability to understand intra-sentential CS chatbot interactions when they are fine-tuned using in-domain monolingual corpora in both CS languages.

6 Conclusions and Future Work

The main contribution of this work is proving that fine-tuning a mBERT NLU model with in-domain bilingual data enables it to detect intents and en-

tities of intra-sentential CS chatbot interactions with industrial grade robustness. It is essential for the corpora employed in the fine-tuning process to share the same intent and entity labels; with this proviso, the model learns to relate and project text entries from the different languages onto a common label space. As a result, the model represents utterance embeddings of the same meaning but different language in the same area of the vector space. Hence, unlike the original mBERT model that groups embeddings into clusters depending on their language, the NLU model fine-tuned on the bilingual in-domain corpus of chatbot interactions happens to be language agnostic, classifying utterances by their meaning regardless of language. This property endows the model with the ability to correctly classify utterances with intra-sentential CS. Given the scarcity of annotated CS data, this outcome is very promising. Considering the results achieved, we strongly recommend exploiting in-domain bilingual corpora to fine-tune mBERT NLU models of real-world chatbot applications in CS regions. Such type of corpora can be easily compiled from the collection of annotated text samples used to train monolingual NLU models without CS, as it has been done in this work.

Overall, it can be stated that progress has been made towards building multilingual conversational assistants that incorporate CS strategies and that can therefore better understand multilingual users. The results obtained for CS between Basque and Spanish, languages belonging to different families, should in principle also be extrapolated to other language pairs.

A tentative line of future research would be to try to further improve performance using data augmentation techniques. To this end, methods that recombine the sentences of the Basque and Spanish corpora while maintaining their semantic labels could be explored. An alternative line of research would be to explore whether the performance of mBERT models fine-tuned on multiple languages sharing utterance labels improves proportionally to the number of languages they are trained with. This would require translating the training corpora to other languages and revising their labelling a posteriori.

References

Gustavo Aguilar, Sudipta Kar, and Thamar Solorio. 2020. Lince: A centralized benchmark for linguis-

tic code-switching evaluation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1803–1813.

Maia Aguirre, Laura García-Sardiña, Manex Serras, Arianne Méndez, and Jacobo López. 2022. *Basco: An annotated basque-spanish code-switching corpus for natural language understanding*. In *Proceedings of the Language Resources and Evaluation Conference*, pages 3158–3163, Marseille, France. European Language Resources Association.

Emily Ahn, Cecilia Jimenez, Yulia Tsvetkov, and Alan W Black. 2020. What code-switching strategies are effective in dialog systems? In *Proceedings of the Society for Computation in Linguistics 2020*, pages 213–222.

Noor Al-Qaysi and Mostafa Al-Emran. Code-switching usage in social media: A case study from oman. *International Journal of Information Technology and Language Studies*, 1(1).

Suman Banerjee, Nikita Moghe, Siddhartha Arora, and Mitesh M Khapra. 2018. A dataset for building code-mixed goal oriented conversation systems. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3766–3780.

Anshul Bawa, Pranav Khadpe, Pratik Joshi, Kalika Bali, and Monojit Choudhury. 2020. Do multilingual users prefer chat-bots that code-mix? let’s nudge and find out! *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–23.

Fengyu Cai, Wanhao Zhou, Fei Mi, and Boi Faltings. 2022. Slim: Explicit slot-intent mapping with bert for joint multi-intent detection and slot filling. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7607–7611. IEEE.

Steven Cao, Nikita Kitaev, and Dan Klein. 2019. Multilingual alignment of contextual word representations. In *International Conference on Learning Representations*.

Giuseppe Castellucci, Valentina Bellomaria, Andrea Favalli, and Raniero Romagnoli. 2019. Multi-lingual intent detection and slot filling in a joint bert-based model.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling.

Ethan A Chi, John Hewitt, and Christopher D Manning. 2020. Finding universal grammatical relations in multilingual bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5564–5577.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 32:7059–7069.

- Björn Gambäck and Amitava Das. 2016. Comparing the level of code-switching in corpora. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1850–1855.
- John J Gumperz. 1982. *Discourse strategies*. Cambridge University Press.
- Yaoshiang Ho and Samuel Wookey. 2019. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813.
- 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 generalisation. In *International Conference on Machine Learning*, pages 4411–4421. PMLR.
- Simran Khanuja, Sandipan Dandapat, Anirudh Srinivasan, Sunayana Sitaram, and Monojit Choudhury. 2020. Gluecos: An evaluation benchmark for code-switched nlp. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3575–3585.
- Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.
- Jitin Krishnan, Antonios Anastasopoulos, Hemant Purohit, and Huzefa Rangwala. 2021. Multilingual code-switching for zero-shot cross-lingual intent prediction and slot filling. In *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pages 211–223.
- Jindřich Libovický, Rudolf Rosa, and Alexander Fraser. 2020. On the language neutrality of pre-trained multilingual representations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1663–1674.
- Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. 2020. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8433–8440.
- Petr Lorenc. 2021. Joint model for intent and entity recognition.
- Meryem M’hamdi, Doo Soon Kim, Franck Dérnoncourt, Trung Bui, Xiang Ren, and Jonathan May. 2021. X-metra-ada: Cross-lingual meta-transfer learning adaptation to natural language understanding and question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3617–3632.
- Telmo Pires, Eva Schlinger, and Dan Garrette. 2019. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4996–5001.
- Libo Qin, Xiao Xu, Wanxiang Che, and Ting Liu. 2020. Agif: An adaptive graph-interactive framework for joint multiple intent detection and slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1807–1816.
- Anirudh Srinivasan. 2020. Msr india at semeval-2020 task 9: Multilingual models can do code-mixing too. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 951–956.
- Marc Tanti, Lonneke van der Plas, Claudia Borg, and Albert Gatt. 2021. On the language-specificity of multilingual bert and the impact of fine-tuning.
- G Richard Tucker. 2001. A global perspective on bilingualism and bilingual education. *GEORGETOWN UNIVERSITY ROUND TABLE ON LANGUAGES AND LINGUISTICS 1999*, page 332.
- Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck. 2010. What is left to be understood in atis? In *2010 IEEE Spoken Language Technology Workshop*, pages 19–24. IEEE.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- 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*.
- Genta Indra Winata, Samuel Cahyawijaya, Zihan Liu, Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2021. Are multilingual models effective in code-switching? *NAACL 2021*, page 142.

Calibrating Imbalanced Classifiers with Focal Loss: An Empirical Study

Cheng Wang, Jorge Balazs, Gyuri Szarvas

Patrick Ernst, Lahari Poddar, Pavel Danchenko

Amazon

{cwngam, jabalazs, szarvasg, peernst, poddar1, danchenk}@amazon.com

Abstract

Imbalanced data distribution is a practical and common challenge in building machine learning (ML) models in industry, where data usually exhibits long-tail distributions. For instance, in virtual AI Assistants, such as Google Assistant, Amazon Alexa and Apple Siri, the *play music* or *set timer* utterance is exposed to an order of magnitude more traffic than other skills. This can easily cause trained models to overfit to the majority classes, categories or intents, leading to model miscalibration. The uncalibrated models output unreliable (mostly overconfident) predictions, which are at high risk of affecting downstream decision-making systems. In this work, we study the calibration of models in the practical application of predicting *product return reason codes* in customer service conversations of an online retail store; The returns reasons also exhibit class imbalance. To alleviate the resulting miscalibration in the trained ML model, we streamline the model development and deployment using *focal loss* (Lin et al., 2017). We empirically show the effectiveness of model training with focal loss in learning better calibrated models, as compared to standard cross-entropy loss. Better calibration, in turn, enables better control of the precision-recall trade-off for the trained models.

1 Introduction

Building and developing ML models in industry has many practical challenges. Imbalanced data distributions (He and Garcia, 2009), particularly long-tail distributions (Wang et al., 2021a), make models overfit to majority data classes and lead to miscalibration (Guo et al., 2017; Mukhoti et al., 2020), i.e. the model-predicted probability fails to estimate the likelihood of true correctness and provides over- or under-confident predictions. To address imbalanced data, some mainstream strategies are rebalancing the dataset through upsampling minority and/or downsampling majority classes.

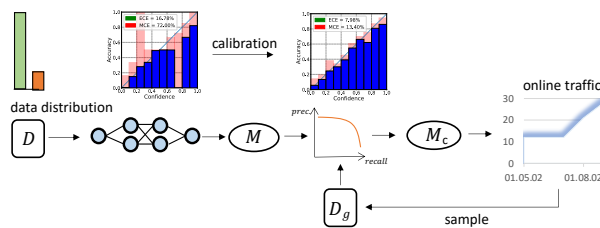


Figure 1: Our procedure to calibrate the ML model that trained with an imbalanced dataset D . With focal loss, model M is learned to be a better calibrated model M_c . In a follow-up stage, a dataset D_g is sampled from actual conversations (the inference distribution) and manually annotated as a golden dataset. It is used to further calibrate models and tune precision-recall threshold to achieve specific precision or recall to serve customer requests.

See SMOTE (Chawla et al., 2002) for an example. However, miscalibration caused by imbalanced data cannot be easily handled by these methods. Guo et al. (Guo et al., 2017) recently found that negative log likelihood (NLL) trained deep neural networks (DNNs) tend to be poorly calibrated as compared to traditional ML methods (Niculescu-Mizil and Caruana, 2005a).

Calibrating models that have been trained with imbalanced data plays an important role in industry. In applications such as medical diagnosis (Caruana et al., 2015), decisions do not only depend on a predicted class, but also on the predicted probabilities, e.g. to quantify patients' risks (Caruana et al., 2015) in order to determine appropriate treatments. Therefore, it is of particular importance that models are well calibrated in high-risk domains such as medicine, financial management (Fischer and Krauss, 2018), self-driving cars (Bojarski et al., 2016), etc., to assure that the predicted probabilities reflect the true values.

In this work, we empirically study the effectiveness of focal loss (Lin et al., 2017) in building reliable ML models for a practical text classification

task – *return reason code prediction* in customer service chatbots. Formally, focal loss is:

$$\mathcal{L}_f = - \sum_{i=1}^N (1 - p_{i,y_i})^\gamma \log p_{i,y_i} \quad (1)$$

where p_{i,y_i} is predicted probability of the i -th sample and γ is a hyper-parameter that is usually set to $\gamma = 1$.

1.1 Practical Considerations

(1) *Theoretical Effectiveness*. Focal loss was originally proposed to handle the issue of imbalanced data distribution which is frequently observed in industrial data (Kilkki, 2007). In learning models, the majority class samples dominate the optimization and gradient descent to update weights in the direction where models become more confident in predicting the majority class. Focal loss can be interpreted as a trade-off between minimizing Kullback–Leibler (KL) divergence and maximizing the entropy, depending on γ (Mukhoti et al., 2020)¹:

$$\mathcal{L}_f \geq \text{KL}(q \parallel p) + \underbrace{\mathbb{H}(q)}_{\text{constant}} - \gamma \mathbb{H}(p) \quad (2)$$

The rationale behind the equation is that we learn a probability p to have a high value (confident) due to the KL term, but not too high (overconfident) due to the entropy regularization term (Pereyra et al., 2017).

(2) *Computational and Algorithmic Complexity*. Out of many popular calibration methods such as temperature scaling (Platt et al., 1999), Bayesian methods (Maddox et al., 2019), label smoothing (Müller et al., 2019) and kernel-based methods (Kumar et al., 2018), focal loss neither increases computational overhead nor requires architectural modifications. For example, the widely used temperature scaling (Platt et al., 1999) requires additional post-training calibration while focal loss offers in-training implicit calibration (by using eq. (2)).

In section 4, we will empirically show the intriguing properties of focal loss in calibrating the trained ML models. Our contributions are summarized as follows:

- We empirically examine the effectiveness of using focal loss in handling model miscalibration in a practical application setting.

- We show that good calibration is important to achieve a desired precision or recall target by tuning the classification thresholds. The standard cross-entropy loss is incapable of achieving this goal due to a skewed predicted probability distribution.
- We demonstrate the performance of calibrated models through a chatbot that serves customers’ requests across three conversational bot use-cases.

2 Background and Preliminaries

2.1 Background

We consider the task of automatic classification of return reason codes in an online retail store, to showcase the development and deployment of ML models. Whenever a customer requests to return a purchased item, a reason code is determined to select the most appropriate resolution and process a return.

For instance, if a customer is not satisfied with the item (its size, color or material, for example) this would map to the return reason *Customer Preference*. In such a case, the appropriate resolution is to process a return and issue a refund, while replacement with the same item is not appropriate (as the customer would face the same issue). In our case study, we consider two use cases: binary reason code prediction and multi-class reason code prediction, where we consider 5 different categories.

2.2 Preliminaries

In this work, we consider the calibration of supervised binary and multi-class classifiers that are trained with imbalanced datasets.

2.2.1 Model Calibration

Calibration (Guo et al., 2017) measures and verifies how the predicted probability estimates the true likelihood of correctness. Assume a model \mathbf{m} trained with dataset $\{\mathbf{x}, y\}, \mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}$. $\hat{\mathbf{p}}$ is the predicted softmax probability. If \mathbf{m} makes 100 independent predictions, each with confidence $p = \arg \max(\hat{\mathbf{p}}) = 0.9$, ideally, a calibrated \mathbf{m} approximately gives 90 correct predictions. Formally, $\text{accuracy}(\mathbf{m}(D)) = \text{confidence}(\mathbf{m}(D))$ if \mathbf{m} is perfectly calibrated on dataset D . It is difficult to achieve perfect calibration in practice.

¹More theoretical findings can be found in the paper

2.2.2 Reliability Diagrams

Reliability Diagrams (DeGroot and Fienberg, 1983; Niculescu-Mizil and Caruana, 2005b) visualize whether a model is over- or under-confident by grouping predictions into bins according to their prediction probability. Predictions are grouped into N interval bins (each of width $1/N$) and the accuracy of samples y_i wrt. to the ground truth label \hat{y}_i in each bin b_n is computed as:

$$\text{acc}(b_n) = \frac{1}{I_n} \sum_i^{I_n} \mathbb{1}(\hat{y}_i = y_i) \quad (3)$$

where $I_n = |b_n|$ i.e. the number of elements in b_n . Let \hat{p}_i be the probability for sample y_i , then average confidence is defined as

$$\text{conf}(b_n) = \frac{1}{I_n} \sum_i^{I_n} \hat{p}_i. \quad (4)$$

A model is perfectly calibrated if $\text{acc}(b_n) = \text{conf}(b_n), \forall n$ and in a diagram the bins would follow the identity function. Any deviation from this represents a miscalibration.

2.2.3 Expected Calibration Error (ECE)

ECE (Naeini et al., 2015) is a scalar summary statistic of calibration. It computes the difference between model accuracy and confidence as a weighted average across bins,

$$\text{ECE} = \frac{1}{I} \sum_{n=1}^N I_n |\text{acc}(b_n) - \text{conf}(b_n)|, \quad (5)$$

where I is the total number of samples.

2.2.4 Maximum Calibration Error (MCE)

MCE (Naeini et al., 2015) measures the worst-case deviation between accuracy and confidence,

$$\text{MCE} = \max_{n \in \{1, \dots, N\}} |\text{acc}(b_n) - \text{conf}(b_n)|. \quad (6)$$

and is particularly important in high-risk applications where reliable confidence measures are absolutely necessary. For a perfectly calibrated classifier, both ECE and MCE are equal to 0.

3 Datasets and Implementation Details

We use historical logged data on return reasons for past human-customer service interactions (which is sometimes noisy), and use human annotated reliable data for model calibration before deployment.

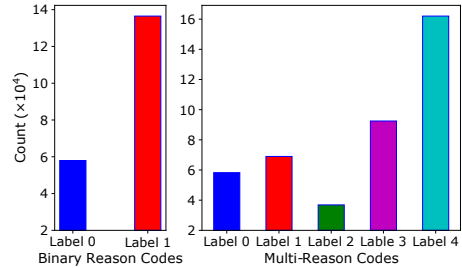


Figure 2: The distribution of randomly sampled datasets for experiments, clearly, we can see the imbalanced label distribution.

Concretely, we prepared 1013 human annotated samples as a golden dataset for the binary reason code model, and a set of 1839 annotated samples for the multi-reason code model. The numbers reported in the tables in Section 4 are based on the annotated dataset.

Figure 2 gives the statistics of randomly sampled datasets from historical logs. We considered 4 widely used return reasons in online retail stores (for details see link below²), subsequently referred as “item is defective or does not work” (LABEL 0), “missing parts or accessories” (LABEL 1), “performance or quality not adequate” (LABEL 2), and “customer preference” (LABEL 3). In the binary case, we aim to detect one particular return type LABEL 0, and discriminate it from any other return reason (referred as OTHERS (LABEL 1)). Similarly in the multi-class case, we consider the four broad return categories (LABELS 0 . . . 3) and gloss all other return types under OTHERS (LABEL 4). In both settings OTHERS is the most frequent class. As we can see the label distribution in both use-cases is imbalanced. For a full dataset D , we split it into $\{D_{train}, D_{val}, D_{test}\}$ with 8:1:1 into train/val/test splits. We train models with standard cross-entropy (CE) and focal loss with different γ values (FL $_{\gamma}$).

We implemented our models with Py-Torch (Paszke et al., 2019), each model consists of 2 bidirectional LSTM layers and 2 dense layers. The embedding dimension is 1024. The hidden layer dimension is set to 128 and 512 for the binary and multi-reason code models respectively. We apply dropout with rates of 0.1 and 0.2 to the embedding and dense layers respectively. The models are trained with Adam (Kingma and Ba,

²<https://www.amazon.de/-/en/gp/help/customer/display.html?nodeId=G6E3B2E8QP HQ88KF>

2014) as the optimization algorithm

4 Model Selection and Calibration

This section describes how we build a model which is well calibrated while retaining its original predictive performance, e.g., 85% precision or 90% recall. We adapt the focal loss (Lin et al., 2017) and empirically evaluate its calibration effectiveness (Mukhoti et al., 2020) in both binary and multi-class text classification tasks in a practical setting.

We observed the two types of trade-offs which are crucial for using models in practice: (1) discrimination-calibration trade-off and (2) precision-recall trade-off. We also found that the value of γ in focal loss plays an important role in learning better calibrated models. We denote the models trained with cross-entropy as CE and with focal loss, for a given value of γ , as FL_γ .

4.1 Discrimination-Calibration Trade-Off

A calibration method should be predictive (discriminative in our case) performance preserving (Zhang et al., 2020).

4.1.1 Binary Reason Code Prediction

Table 1 presents the results of models with different loss functions. Note that the CE based model achieves the best predictive performance while FL-based models give slightly lower scores. However, FL performs significantly better than CE in terms of calibration related metrics (i.e., NLL, ECE and MCE). We can also observe that the higher the γ value the better calibration performance on human annotated samples.

Figure 3 presents the predicted softmax probability distribution (top) as well as the reliability diagram (bottom). From (a)-(e), we can clearly see that probabilities change from a spiking distribution (overconfident: p is close to either 1 or 0) to a flatter distribution, for instance, $p = \{0.6, 0.4\}$. Figure 3 (f)-(j) show that a miscalibrated model exhibits high ECE and MCE scores. We can also observe this miscalibration through the gaps between confidence and true likelihood of correctness. In this binary imbalance classification case, we find that calibration slightly hurts predictive performance but it is modest. The best model can be obtained by selecting the candidate that achieves the best discrimination-calibration trade-off. Here it should be FL_5 according to Table 1.

4.1.2 Multi-Reason Code Prediction

We further conducted multi-reason code prediction experiments covering 5 reason codes. Table 2 and Figure 4 demonstrate the effectiveness of focal loss in learning well calibrated models. It is important to note that preparing a small set of human labeled samples as a golden dataset is crucial to measure whether a model is calibrated or not. The golden dataset reflects the online data distribution which our model will predict after deployment. The difference in calibration effects can be observed by comparing Figure 4 (top row, (a)-(c)).

4.2 Precision-Recall Tradeoff

The trade-off between model precision and recall (Buckland and Gey, 1994) is an important aspect in deploying trained models. For instance, if a certain prediction task requires high model precision (recall), it means recall (precision) needs to be sacrificed to some extent. The trade-off can be achieved through classification threshold tuning. In this subsection, we empirically demonstrate that a better calibrated model can help to accomplish this goal.

Figure 5 presents the precision-recall curves for binary reason code models. The corresponding softmax probability distribution is shown in Figure 3 (top). For CE model, Figure 5 (a) and Figure 3 (a), we found that it gives more polarized probability, i.e., the predicted probability is more spiky. Given this skewed distribution, it is difficult to tune a precision based on a given recall, or a certain recall based on an expected precision, e.g., 85%. On the other hand, FL models in Figure 5 (b-d) learn a flattened probability distribution, which is better distributed across the $[0, 1]$ interval and is therefore more amenable to thresholding for a particular precision (or recall) target. This behavior can be also observed in Figure 6 when we compare the computed thresholds in CE and FL models.

5 Experimental Results

To serve customers' requests, we use the binary reason code model for three conversational use-cases³. The model detects whether a product return

³This empirical study was conducted using the model trained with focal loss, $\gamma = 1$ (FL_1 in Table 1). Thus it was not the best model variant according to our findings. Note that to minimize risk and maintain customer experience, we don't deploy and compare multiple models online, at same time. However, the superiority of FL-based model as compared to CE-based model is clearly observed from offline results in previous sections.

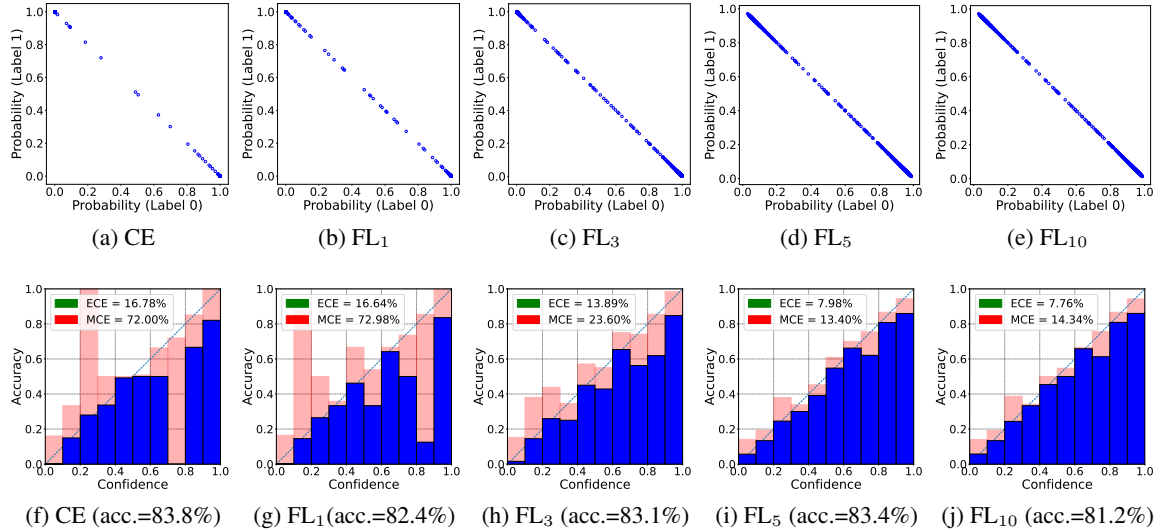


Figure 3: The reliability diagram plots for binary-reason code models with 10 bins. The diagonal dash line presents perfect calibration (on a specific bin, confidence is equal to accuracy.)

Metric	CE	FL ₁	FL ₃	FL ₅	FL ₁₀
Accuracy	0.836	0.824	0.831	<u>0.834</u>	0.816
Precision	0.838	0.822	0.830	<u>0.834</u>	0.807
Recall	0.823	0.814	0.821	0.823	0.805
F1	0.828	0.817	0.824	<u>0.827</u>	0.806
NLL	2.159	1.438	0.608	<u>0.258</u>	0.178
ECE	0.168	0.166	0.139	<u>0.080</u>	0.078
MCE	0.720	0.730	0.236	<u>0.134</u>	0.143

Table 1: The performance of binary reason code models with CE and Focal loss (with different γ values) on 1013 human annotated samples. For predictive performance (top rows e.g., accuracy) the differences across models are negligible. However, on the calibration related metrics (bottom rows), FL-based models show significantly better performance. The best scores are marked **bold** and the second best scores are underlined, same as Table 2

Metric	CE	FL ₁	FL ₅
Accuracy	0.751	0.760	0.751
Precision	0.814	<u>0.807</u>	0.814
Recall	0.757	<u>0.755</u>	0.757
F1	0.764	<u>0.760</u>	0.764
NLL	0.599	<u>0.429</u>	0.309
ECE	<u>0.037</u>	0.023	<u>0.037</u>
MCE	<u>0.296</u>	0.197	0.299

Table 2: The performance of multi-reason code models with CE and focal losses (with different γ values).

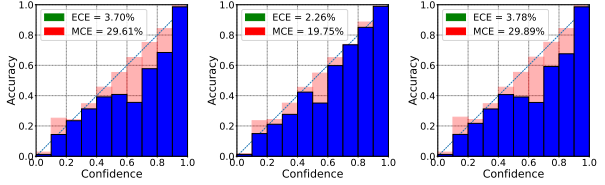
is a special case LABEL \emptyset according customers' free-text input. We analyze the performance of the model through both intrinsic evaluation of model's accuracy of predicted reason code and extrinsic evaluation on a downstream conversational chatbot system. Before deployment, we tuned model to achieve expected 85% precision with thresh-

old=0.512 for LABEL \emptyset .

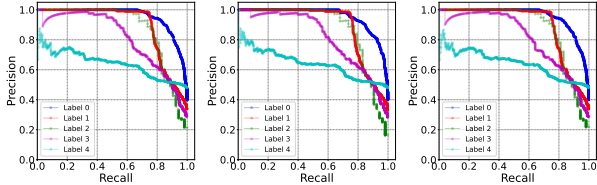
5.1 Intrinsic Evaluation

We conduct a human evaluation of the model's prediction on a sample of contacts from actual conversations that the model has served. First, we randomly sample 485 contacts where the reason code model predicted the LABEL \emptyset code. By manually annotating those contacts, we found 384/485 to be correctly predicted, i.e. the model achieves a precision of 83.8% after deployment. This aligns well with the precision (81.4%) computed on the offline test set.

Although we are primarily interested in the precision metric, we also analyze the negative predictive value of the model for opportunity analysis of future improvements. We randomly sampled 200 contacts where the model *did not predict* the LABEL \emptyset code. Out of 200 contacts, we find 194 predic-



(a) CE (acc.=75.1%)(b) FL₁(acc.=76.0%)(c) FL₅ (acc.=75.1%)



(d) CE (f1=76.4%) (e) FL₁(f1=76.0%) (f) FL₅ (f1=76.4%)

Figure 4: The reliability diagram plots for multi-reason code models with 10 bins. The diagonal dash line presents perfect calibration (on a specific bin, confidence is equal to accuracy.)

tions are true, i.e. the model demonstrates a high precision of $194/200 = 97\%$ for OTHERS .

5.2 Extrinsic Evaluation

To evaluate the impact of our model in a downstream application, we run an online A/B experiment, with and without the presented model, for a conversational bot in three use-cases: (1) *control branch*: conversational bot with customer selected reason code; (2) *treatment branch*: conversational bot with additional check for return reason code (LABEL 0) in customer free text input.

To evaluate a conversational bot, we measure the following three key metrics: (1) *Automation Rate* (AR): The % of contacts that were resolved by the conversational bot without requiring any human involvement; (2) *Positive Response Rate* (PRR): The rate measures the % of times customers responded positively to the resolution provided by chatbot; (3) *Repeat Rate* (24RR): The rate of customer contact us again for the same issue in 24 hours.

Table 3 presents the results on these key metrics over a period of two weeks. As can be observed from the results, the reason code model improves the performance of the conversational system on both automation rate and customer experience related metrics. This is achieved through enabling the bot to provide appropriate solutions based on the specific customer situations.

Applications	Metrics	C(%)	T(%)
use-case A	AR	—	+ 2.13
	PRR	—	+ 3.18
	24RR	—	- 0.65
use-case B	AR	—	+ 2.10
	PRR	—	+ 0.97
	24RR	—	- 0.68
use-case C	AR	—	+ 3.98
	PRR	—	+12.85
	24RR	—	- 1.02

Table 3: Online evaluation of our model for three applications (top, middle, bottom). The relative numbers are reported. AR and PRR, the higher the better. 24RR, the lower the better. We only report performance relative numbers due to confidential issues. C: Control branch; T: Treatment branch.

6 Related Work

Guo *et al.* (Guo *et al.*, 2017) and Mukhoti *et al.* (Mukhoti *et al.*, 2020) showed that the miscalibration of larger, modern networks is related to the over-fitting on the likelihood of the training dataset. Conventional NNs are trained to minimize the negative log likelihood (NLL) which can be positive even if the accuracy is already perfect. Modern networks continue to minimize NLL during training after accuracy is optimal, overfitting to the training dataset and becoming increasingly confident in incorrect predictions as a result. To tackle this, researchers propose a variety of regularizers for model predicted probability including the focal loss (Lin *et al.*, 2017), acting as a maximum entropy regularizer (Mukhoti *et al.*, 2020), and temperature scaling (Guo *et al.*, 2017). Muller *et al.* (Müller *et al.*, 2019) suggest using label smoothing to regularize network outputs. Besides, some recent methods, such as Bayesian method (Maddox *et al.*, 2019), meta-learning (Bohdal *et al.*, 2021), Gumbel-softmax Trick (Jang *et al.*, 2017; Wang *et al.*, 2021b) and kernel-based method (Kumar *et al.*, 2018) are proposed to learn better calibrated model directly from training.

7 Discussion

When calibrating models in practical scenarios, the complexity of calibration is an issue to consider. Focal loss (Lin *et al.*, 2017; Mukhoti *et al.*, 2020) offers in-training calibration via entropy regularization (Pereyra *et al.*, 2017). In this work we have shown analytically (Section 4) that it provides a

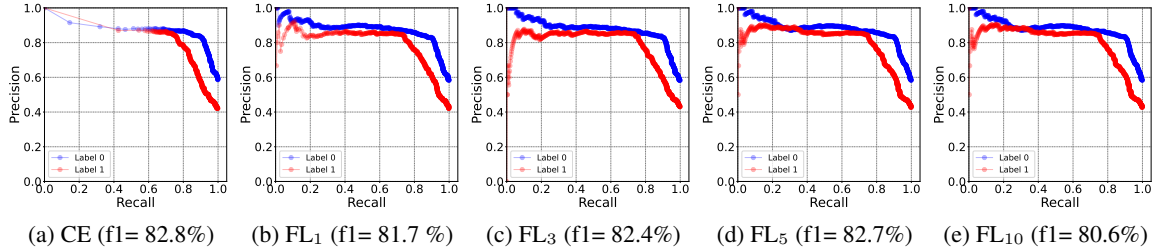


Figure 5: The precision-recall curves for binary reason code models. (a) CE model gives more polarized probability and makes it difficult to tune a precision based on a given recall or vice versa. (b-d) FL learns to give better distributed probability, precision or recall can be tuned more easily.

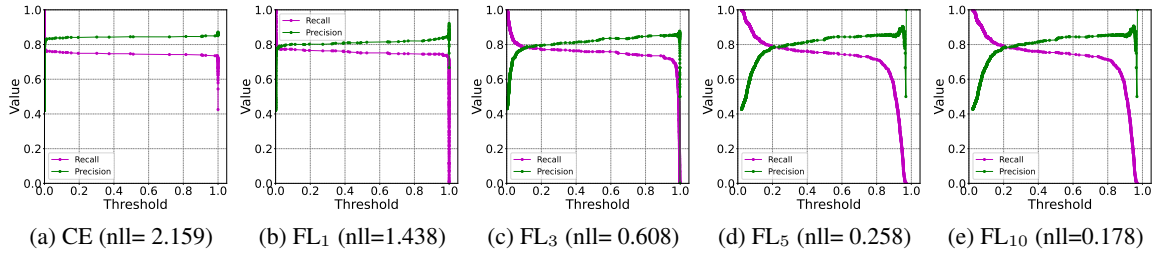


Figure 6: The curves of precision and recall against threshold in binary reason code models.

simple and effective way to calibrate a trained ML model. On the other hand, in some cases we need to tune γ value for datasets. Experimentally, we found that setting a high γ value would not significantly hurt predictive performance while providing good calibration performance.

8 Conclusion

In this paper, we empirically showed the effectiveness of using focal loss in learning better calibrated models and finding the precision-recall trade-off in practical application of deep neural network models. We conducted an in-depth analysis of miscalibration caused by imbalanced data distribution and the existing issues of using cross-entropy trained models in practical settings. We also showed that the hyperparameter γ , which theoretically controls the entropy regularization term, is important to model calibration. We studied the deployment of an ML model in practical use cases and demonstrated that better calibration helps to control the precision-recall trade-off through posterior thresholding and improves post-deployment metrics (in an online A/B test).

Ethical Considerations

Development and Experiments. We used anonymized text dialogue snippets to train the mod-

els. The particular model described in this work has no way to reveal customer information. We do not release the datasets used in the experiments.

Failure Modes. Regarding risks related to system errors, incorrect predictions of the models described in this work may result in wrong return reason assignment. However, the practical risk related to such misclassification is limited, because the customers interacting with the chatbot have an option to talk to a human associate if they consider the system doesn't work as expected.

References

- Ondrej Bohdal, Yongxin Yang, and Timothy Hospedales. 2021. Meta-calibration: Meta-learning of model calibration using differentiable expected calibration error. *arXiv preprint arXiv:2106.09613*.
- Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Praseem Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.
- Michael Buckland and Fredric Gey. 1994. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19.
- Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 2015. Intelligent models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730.
- Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.
- Morris DeGroot and Stephen Fienberg. 1983. The comparison and evaluation of forecasters. *The Statistician*.
- Thomas Fischer and Christopher Krauss. 2018. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669.
- 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. PMLR.
- Haibo He and Edwardo A. Garcia. 2009. [Learning from imbalanced data](#). *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284.
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparametrization with gumble-softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net.
- Kalevi Kilkki. 2007. A practical model for analyzing long tails. *First Monday*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. 2018. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2805–2814. PMLR.
- 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.
- Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. *Advances in Neural Information Processing Systems*, 32.
- Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. 2020. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299.
- Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? *Advances in neural information processing systems*, 32.
- Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*.
- Alexandru Niculescu-Mizil and Rich Caruana. 2005a. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632.
- Alexandru Niculescu-Mizil and Rich Caruana. 2005b. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*.
- 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 *NeurIPS’19*, pages 8024–8035.
- Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *arXiv preprint arXiv:1701.06548*.
- 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.
- Cheng Wang, Sun Kim, Taiwoo Park, Sajal Choudhary, Sunghyun Park, Young-Bum Kim, Ruhi Sarikaya, and Sungjin Lee. 2021a. Handling long-tail queries with slice-aware conversational systems. In *ICLR 2021 Workshop on Weakly Supervised Learning*.
- Cheng Wang, Carolin Lawrence, and Mathias Niepert. 2021b. Uncertainty estimation and calibration with finite-state probabilistic rnns. In *ICLR*.

Jize Zhang, Bhavya Kailkhura, and T Yong-Jin Han. 2020. Mix-n-match: Ensemble and compositional methods for uncertainty calibration in deep learning. In *International conference on machine learning*, pages 11117–11128. PMLR.

Unsupervised training data reweighting for natural language understanding with local distribution approximation

Jose Garrido Ramas

Amazon Alexa AI

jrramas@amazon.com

Dieu-Thu Le

Amazon Alexa AI

deule@amazon.com

Bei Chen

Amazon Alexa AI

chenbe@amazon.com

Manoj Kumar

Amazon Alexa AI

abithm@amazon.com

Kay Rottmann

Amazon Alexa AI

krrothm@amazon.com

Abstract

One of the major challenges of training Natural Language Understanding (NLU) production models lies in the discrepancy between the distributions of the offline training data and of the online live data, due to, e.g., biased sampling scheme, cyclic seasonality shifts, annotated training data coming from a variety of different sources, and a changing pool of users. Consequently, the model trained by the offline data is biased. We often observe this problem especially in task-oriented conversational systems, where topics of interest and the characteristics of users using the system change over time. In this paper we propose an unsupervised approach to mitigate the offline training data sampling bias in multiple NLU tasks. We show that a local distribution approximation in the pre-trained embedding space enables the estimation of importance weights for training samples guiding resampling for an effective bias mitigation. We illustrate our novel approach using multiple NLU datasets and show improvements obtained without additional annotation, making this a general approach for mitigating effects of sampling bias.

1 Introduction

Production Natural Language Understanding (NLU) models are typically trained on the offline annotated data. Models learn from the offline data to perform classification on the online live data in production after the model being deployed.

The core of voice-controlled assistants, such as *Google Home*, *Amazon Alexa*, or *Siri*, apply NLU models to perform both *intent classification* and *slot labelling* (Weld et al., 2021). For example, the input utterance "*set alarm at 9 am*", would be classified as "*SetAlarmIntent*" intent, and the slots "*9*" and "*am*" would be labelled as *Time*.

In the deployed NLU systems, a distribution mismatch between training and live data is common. Some factors contributing to such a mismatch are

changes of the live data distribution over time (due to, for example, new users or to seasonal changes), and usage of data from other more or less unrelated tasks to enrich the training data, so called out-of-domain data.

The issue with this mismatch in distribution between training and inference time is that models learn a bias towards specific classifications that is not existing at inference time. Even if the label distributions are matched, it is still possible that the model will have biased performance, since demographic and speech differences need not perfectly correlate with label distribution, resulting in degraded accuracy and possibly unequal performance across populations (Subramanian et al., 2021). Thus, mitigation of this distribution mismatch is an important step in the development of models.

While a common approach of dealing with this kind of bias is manual upsampling of classes in the training data (Estabrooks et al., 2004), this approach is not always optimal, due to the complexity and variation of natural language. Data of the same class in a classification task often come from very different forms of language, for example slang vs. formal language. A simple upsampling based on the classes does not mitigate differences in usage of slang during inference time compared with the training data.

Another difficulty in class based resampling is to get the correct label distribution from the live data in the case that it is missing ground-truth. For example, when a model is deployed, its training data will ideally match the distribution of the current live data, since this is the data the model will be applied to. This current live data distribution will be more similar to recent live data, compared to historical data. However, manually annotating data to obtain the ground-truth labels takes time. Thus, during deployment, the training data should match the unannotated live data, and in this case it is only

feasible to use bias reduction methods which don't rely on manual annotation.

In this work we build on top of importance weighting, which is an approach that has gained traction in other machine learning fields, but until now has found little attention on natural language understanding. We propose a method to assign weights to every individual utterance in a training corpus based on observed live usage of the system, by using utterances' *neighbourhood* in the embedding space. We choose to find the *neighbourhood* of utterances with KNN and KMeans (in the case of KMeans, each cluster is considered a neighbourhood). We choose these methods due to their easy interpretation: For example, in the KMeans case, one can observe a cluster of utterances, its frequency online and offline, and easily understand why this specific pattern has a high or small weight.

The two unsupervised re-weighting based on KNN and KMeans are compared with two baselines: keeping the training data as it is (with the distribution mismatch), and, on the other hand, a semi-supervised intent-based approach, in effect class up/down-sampling. We evaluate our methods on both public data and in a deployed commercial NLU system. In the public datasets, we simulate a distribution mismatch by both introducing a label mismatch and also combining different sources of data with different distributions.

We show that the unsupervised approaches can better mitigate certain kinds of sampling bias compared to the intent-based approach, while also having the advantage that we can perform re-weighting of the training data without need of annotation: thus our method is suitable for test data with fast-changing distribution. Without the need for any labeled data, our unsupervised approaches are generic enough to be applicable to multiple different natural language processing tasks.

2 Related Work

The problem of dealing with training data sampling bias in machine learning is well studied. The idea of adjusting training data distribution to meet the distribution at inference time is discussed in (Zadrozny, 2004), (Shimodaira, 2000) and (Dudík et al., 2005). These methods however require estimation of biased densities or selection probabilities, which pose a challenge in the real world.

Similarly in (Grover et al., 2019), to deal with

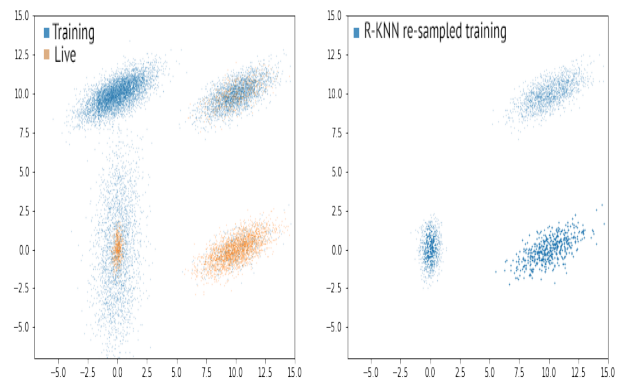


Figure 1: Example of *training* (blue, left) and live data (orange, left) with different distributions, as well as the output of R-KNN resampling the training data (blue, right). Darker points indicate higher weight.

bias in generative models, a classifier is learned to distinguish the data distribution from the generative model. This allows guidance of the generation of additional data to better mimic the existing data. In this work we extend on the work above towards natural language understanding, and focus on the real world problem in which the training data is biased with respect to the unannotated real world application data (live data).

In (Huang et al., 2006) unsupervised model-agnostic importance weights for every training sample are computed. Our unsupervised approach differs from theirs in that we calculate the weights based on the neighbourhood, which makes interpretation of the individual weights easier in the case of natural language data. A closer investigation of importance weighting can also be found in (Cortes et al., 2010) providing theoretical bounds, as well as in the recent work of (Fang et al., 2020) that looks specifically at the application of importance weighting and weight estimation for deep learning tasks. An important difference to these approaches is that they focus on including importance weighting directly into the learning of the models. In our work we focus however solely on the underlying data distribution of utterances, while keeping the estimation model the same.

In contrast to importance weighting, another common approach in real world applications is the use of pure upsampling of training utterances for certain classes, based on automatic labelling of the live data. In (Estabrooks et al., 2004) the effect of upsampling for certain underrepresented classes

is investigated, showing its effectiveness. On the other hand looking at the class distribution alone will also not reduce data bias as described in Sec 1, making the requirement of an automatic way of handling different kinds of distribution mismatch more pronounced.

3 Utterance Weight Estimation

In this section, we describe our approach on how to estimate the weight of each individual utterance in offline training data based on a random sample from online live data.

Let X represent the random variable of an utterance from online live data, where X follows some distribution P_X , denoted as $X \sim P_X$. Let Y be the random variable of true labels of X , where Y follows some distribution P_Y , denoted as $Y \sim P_Y$. Also, let X' and Y' be the corresponding random variables of X and Y in offline data, where $X' \sim P_{X'}$ and $Y' \sim P_{Y'}$. The issue we aim to resolve is that typically $P_{X'} \neq P_X$ and $P_{Y'} \neq P_Y$.

Analysing the difference in distributions of utterances P_X and $P_{X'}$ is particularly challenging in NLP because the different surface forms of utterances do not necessarily imply the semantic difference in classification tasks. However, due to the advance of natural language embeddings, we are now able to efficiently approximate the local distributions over the semantic meanings of text which allows the estimation of P_X and $P_{X'}$. Specifically, we propose to approximate the difference of local distributions in offline and online utterances summarized as follows:

1. Map all utterances of offline training and online live data into the embedding space.
2. For every offline training utterance x_i , estimate the local approximations of P_X and $P_{X'}$, denoted as \hat{P}_X and $\hat{P}_{X'}$ and compute the weight using its neighbourhood utterances

$$w_i = \frac{\hat{P}_X(X = x_i)}{\hat{P}_{X'}(X' = x_i)}.$$

3. Resample the utterance x_i in offline training data according to the weight w_i .

3.1 Mapping utterances into embedding space

Pre-trained BERT-based models sentence-level representations do not guarantee that semantically similar utterances will be close in the embedding space.

Thus, for the mapping of the text into the embedding space, we use Sentence-BERT (Reimers and Gurevych, 2019a), which modifies the original BERT architecture via siamese and triplet network structures to compute semantically meaningful embeddings which can be compared using several functions such as cosine similarity or euclidean distance.

3.2 Local Distribution Approximation

To mitigate the distribution mismatch we aim to de-bias the local distribution of each training utterance to match the live data distribution. Having embedded utterances into the embedding space in the first step, it is now possible to estimate the local neighbourhood of text utterances by using the distances in the embedding space. Then, we are able to determine an approximation of the local distributions P_X and $P_{X'}$ at some given utterance by looking at the number of samples in this neighbourhood that belong to either X or X' . In the following we propose three different reweighting methods, which differ on how the neighbourhood of each utterance is defined: Reweighting K-Nearest-Neighbour (R-KNN), Reweighting KMeans (R-KMeans) and, as an additional method, an intent-based approach (B-intent; effectively class up/down sampling).

R-KNN (Reweighting via KNN): The first local approximation we discuss is based on k-nearest-neighbours. We follow the standard procedure to use $K = \sqrt{N}$ with N being the total number of utterances in training and the live sample combined.

We aim to determine the weight the individual training utterance x_i , by using a sample of embedded training samples T and of live samples L . Let $\text{KNN}(x)$ be the set of K nearest neighbours to a point x in the embedding space. We determine:

$$D_{train}^{(i)} = \bigcup_{e \in \text{KNN}(x_i), e \in T} e$$

the set of all utterances that are part of both the neighbourhood of a training utterance x_i and the training data. In a similar way we determine the set of all utterances from the live traffic sample L that fall into the neighbourhood of x_i :

$$D_{live}^{(i)} = \bigcup_{e \in \text{KNN}(x_i), e \in L} e$$

With these two sets, we approximate the probability of having a training sample in this region of the

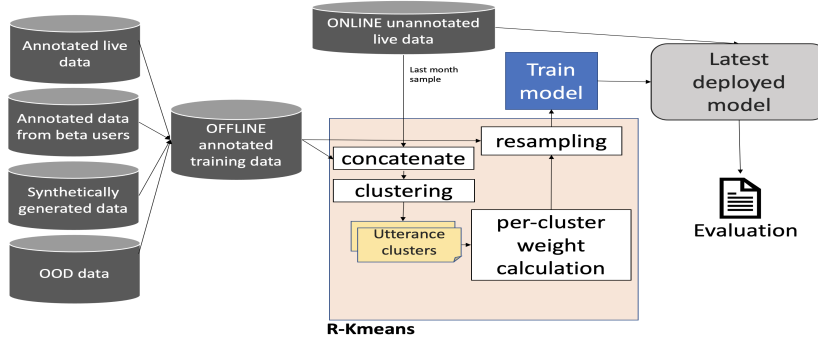


Figure 2: Pipeline for utterance reweighting. We combine many different sources of training data, and then assign a high/low weight to each utterance depending on the recent, unannotated *live* data, which follows the most similar distribution the data the model will be applied to (compared to, for example, historical annotated live data)

embedding space:

$$p(x \in T | \text{neighbourhood}(x_i)) = \frac{|D_{train}^{(i)}|}{|T|}$$

And similarly we approximate the probability of a live utterance x being seen in this region of the embedding space as

$$p(x \in L | \text{neighbourhood}(x_i)) = \frac{|D_{live}^{(i)}|}{|L|}$$

. The ratio of these two probability approximations is the weight we assign to the utterance x_i :

$$w_i = \frac{p(x \in L | \text{neighbourhood}(t_i))}{p(x \in T | \text{neighbourhood}(t_i))}$$

w_i therefore indicates therefore how much more likely it is that an utterance in a certain region is part of the live traffic in comparison to being part of the training data.

R-Kmeans (Reweighting via KMeans) Another way of approximating neighbourhoods is with unsupervised clustering. In this case the training and live data are combined and then clusters are computed in the embedding space. Then, the neighbourhood is all utterances within the same cluster. Thus, all utterances within a cluster obtain the same weight. After having found the neighbourhoods $D_{train}^{(i)}$ and $D_{live}^{(i)}$ through clustering, we follow exactly the same equations as above to compute the weights. For simplicity we chose K-Means clustering (MacQueen et al., 1967) and chose K as $K = \sqrt{N}$. If the live data and the training data came from the same distribution, it would be expected to find that, in each cluster i , $\frac{D_{train}^{(i)}}{D_{live}^{(i)}} \approx \frac{|T|}{|L|}$. After reweighting each utterance with a weight calculated with R-KMeans, the above equality is true on every cluster.

B-intent (Baseline via intent) As a baseline, we reweight the data based on the label distribution. The problem with this approach is it can't address latent distribution mismatches not directly related to the labels, as for example formal and informal language (see Sec 1). We train a classifier on the biased training data to infer \hat{P}_Y , an approximation of P_Y , and we use P_Y^l as is known from the annotated training data. We give each intent a weight as: $w_{intent} = \frac{\hat{P}_Y(intent)}{P_Y^l(intent)}$ which is in line with the description above for R-KNN, considering the neighbourhood of an utterance to be made of all utterances with the same label. As a result after reweighting the utterances of every intent with the weight of their intent w_{intent} , the labels of the resampled data will follow $\hat{P}_Y(intent)$.

3.3 Resampling the Training Data

With the computed weights for every training example, we are now able to resample the training data according to the live data distribution.

A weight < 1.0 means, that this training utterance is less reflective of the live distribution, while a weight > 1.0 reflects utterances more important for matching the live distribution.

While there are different ways in the literature of using this reweighting information, like (Fang et al., 2020) and (Huang et al., 2006) using it directly as part of the optimisation in the learning of the machine learning model, we chose the most straight forward of up- and down-sampling the utterances directly in the training data. A toy example of R-KNN resampling can be seen in Fig. 1.

4 Experiments

In our experiments we evaluated our methods on multiple different NLU datasets to verify the feasi-

	B-Bias	B-intent	R-KNN	R-KMeans	R-intent (rel)	R-KNN(rel)	R-KMeans (rel)
snips int	0.0164	0.0161	0.0162	0.0157	-1.82927	-1.21951	-4.26829
slurp int	0.1542	0.15	0.148	0.1462	-2.72374	-4.02075	-5.18807
snips utt	0.1329	0.13	0.1214	0.1258	-2.18209	-8.65312	-5.34236
slurp utt	0.34	0.3372	0.3324	0.3322	-0.82353	-2.23529	-2.29412

Table 1: Intent ("int") and utterance ("utt") error rates of the different methods in SLURP/ SNIPS datasets. Best result in bold. Each experiment is run ten times, and the average is reported. Both absolute value and relative change with respect to the first baseline is also reported.

bility of the approach.

4.1 Datasets

We tested our methods on a large commercial voice assistant dataset, as well as in two public ones: SLURP (Bastianelli et al., 2020) and SNIPS (Coucke et al., 2018). In all these datasets, the NLU task is intent classification and slot labelling. In the commercial dataset case, data is de-identified.

The training and test data are manually annotated, whereas the live data isn't. In the commercial voice assistant scenario, we take a sample of last month's unannotated live data as representative of current usage of the system. The size of the sample is the same as the offline training data. The annotated live data (test data), is not available during model deployment, but can be obtained afterwards to estimate the performance of the method.

4.2 Bias simulation strategies

Most available natural language understanding datasets are very well curated, with the test sets closely resembling the distribution of the training data. Thus, in the public datasets we simulate bias that could occur in real world applications via two different strategies on the training data:

Intent-based sampling bias: We introduce bias in the label distribution in the following way: each intent is assigned to either a low-sampling bucket (with probability 20%) or to a high-sampling bucket (with probability 80%). The two intents that are in common between SNIPS and SLURP tasks (related to weather and to music) are both assigned to the low-sampling bucket. Finally, intents in the low-sampling buckets are down-sized to 20% of their original size, by randomly removing 80% of utterances which are annotated as belonging to this intent. The high-sampling intents are left as is.

Add OOD data: To introduce bias not directly related to the labels, as well as mimic the real-life scenario in which the training set is composed of different data sources with different amounts of

noise, we also add, to each task, the training data of the other task. That is, we add the SNIPS data to the SLURP training set, and we add the SLURP data to the SNIPS training set. Prior to adding the data, we first produce machine-annotated labels for the SNIPS utterances in the SLURP label space, as well as labels to the SLURP utterances in the SNIPS label space.

4.3 Experimental Setup

The embeddings were generated with *paraphrase-MiniLM-L6-v2* model part of (Reimers and Gurevych, 2019a) sentence transformer model family. This model is fine-tuned so that semantically similar sentences are close in the embedding space with respect to distance functions, including euclidean distance (Reimers and Gurevych, 2019b).

To not leak information of the unseen test data into the reweighting, we used the development data for the distribution approximation.

For the resampling, we upsampled utterances with a weight w_i to frequency: $n_i = \lfloor w_i \rfloor + \theta$, where θ is random variable that is 1 with $p = w_i - \lfloor w_i \rfloor$, and 0 otherwise. The expected value is $E[n_i] = w_i$

We train a BERT model (Chen et al., 2019). For hyperparameter tuning, we follow (Chen et al., 2019), and use adam optimizer (Kingma and Ba, 2014) over 4 epochs, with a learning rate of $5e-5$ and batch size 32. We use the implementation from (Wolf et al., 2019), with *bert-base-uncased* pretrained model. We report f1-score on the test data.

We compare our unsupervised approaches (**R-KNN** and **R-KMeans** from Sec 3.2) with two baselines: **B-Bias** (baseline model trained on the biased data) and **B-Intent**, baseline model in which the biased data is up/down-sampled so that the label distribution matches the live data (see Sec 3.2). The BERT model described above is used to obtain the hypothesised intent on the live data.

4.4 Results

Public datasets: The results of our experiments can be seen in table 1. We report intent classification error rate, as well as utterance error rate. We define utterance error rate as the fraction of utterances in which there is an error either in the slot labelling or intent classification task.

Each experiment is run ten times, and the average error rate is reported. The difference between both R-KMeans and the two baselines (B-intent and B-bias) passes a two-sided paired t-test for statistical significance at 95% confidence level.

The difference Between the R-KMeans and R-KNN approaches is, however, not statistically significant. R-KMeans has the advantage over R-KNN of easier interpretation of the weights: one weight is produced per cluster, instead of per utterance. The clusters can manually be inspected, and, comparing the in-cluster frequency of the *live* and *training* data, understand why this cluster got a high/low weight.

For example, we observe in our SNIPS run two distinct clusters related to weather queries that get different weights: the first one, related to questions about specific weather events (such as snow or rain: includes, for example, the utterance *"is it snowing in California"*). Using **R-KMeans** reweighting, this cluster receives a weight of 1.04 (which can be interpreted roughly as: this pattern of utterances is equally frequent in the live data (development SNIPS data in this case) as in the training data. Thus, it does not need to be upsampled or down-sampled.

However, a different cluster of weather queries containing more general questions *"what is the weather forecast for Akers New Hampshire"* receives a weight of 12: This cluster is 12 times less frequent in the training data than in the live data. Thus, this cluster is upsampled by 12.

Overall, in our experiments the two unsupervised methods perform better than both intent-based resampling and the baseline. A limitation of our work, however, is that it requires live annotated data to use as test data, to estimate the performance post model deployment. Obtaining this data can be a challenge in real-life applications.

Commercial dataset: On the commercial dataset, we show that, in the case that the training data has a different distribution to the *live* and test data, applying reweighting techniques with local distribution approximation can improve

	Intent	Utterance
Overall	-4.63	-1.99
Global	-2.02	-1.29
HomeAutomation	-13.77	-9.49
Knowledge	-2.1	-2.17
ToDos	-12.34	-3.99
Notifications	-9.09	-6.29

Table 2: Relative reduction in error rates (both intent and utterance) in the commercial dataset.

performance. We compare the results of applying reweighting on the training data vs. without reweighting strategy and report the relative differences. We use R-KMeans reweighting, due to the easier manual inspection of the assigned weights (see Sec 3.2).

We report the relative difference in both intent error rate and utterance-error rate. As shown in Table 2, we see improvements in both utterance and intent error metric, with the biggest coming from Home Automation domain (13.77%), and an overall improvement of 4.63% accross all domains. The results with respect to the baseline passes a two-sided paired t-test for statistical significance at 95% confidence level.

5 Conclusion and future work

In this work, we showed how the reweighting of training data using local distribution approximation helps in mitigating sampling bias in natural language understanding production models. We simulated the bias in public training datasets to mimic real world application scenarios in which different data sources are used, and they each come from different distributions. We reweighted utterances based on the approximation of local distribution to minimise the mismatch between the training and online live traffic data. The simplicity of our approach, and the fact that it does not require manual or machine annotation, means that it can be used to quickly adapt the training data to the ever-changing live data in deployed models. Experiments in both a commercial dataset and two public datasets have shown that our approach can mitigate the mismatch and bias in training data without additional manual tuning. In the future, we want to experiment the combined impact of our method with different data augmentation techniques, study the impact on fairness across populations, as well as bias detection methods to trigger the reweighting model.

6 Ethical considerations

In this work we apply a reweighting method before model deployment to mitigate the problem of bias in the training data compared to the live data. We target overall accuracy as the metric we aim to improve, and we achieve so by tailoring the model to the latest live data at model deployment. However, the impact of reweighting on per-population accuracy has not been studied. There is a risk that, due to focusing on current live data, populations which at the time of a model deployment are not extensively using the model are not well-served by the reweighting, even though overall accuracy improves.

References

- Emanuele Bastianelli, Andrea Vanzo, Pawel Swietojanski, and Verena Rieser. 2020. [SLURP: A spoken language understanding resource package](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7252–7262, Online. Association for Computational Linguistics.
- Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.
- Corinna Cortes, Yishay Mansour, and Mehryar Mohri. 2010. Learning bounds for importance weighting. In *Nips*, volume 10, pages 442–450. Citeseer.
- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Calta-girone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.
- Miroslav Dudík, Steven Phillips, and Robert E Schapire. 2005. Correcting sample selection bias in maximum entropy density estimation. *Advances in neural information processing systems*, 18:323–330.
- Andrew Estabrooks, Taeho Jo, and Nathalie Japkowicz. 2004. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36.
- Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. 2020. Rethinking importance weighting for deep learning under distribution shift. *arXiv preprint arXiv:2006.04662*.
- Aditya Grover, Jiaming Song, Alekh Agarwal, Kenneth Tran, Ashish Kapoor, Eric Horvitz, and Stefano Ermon. 2019. Bias correction of learned generative models using likelihood-free importance weighting. *arXiv preprint arXiv:1906.09531*.
- Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. 2006. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608.
- Diederik P. Kingma and Jimmy Ba. 2014. [Adam: A method for stochastic optimization](#). Cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.
- James MacQueen et al. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.
- Nils Reimers and Iryna Gurevych. 2019a. [Sentencebert: Sentence embeddings using siamese bert-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Nils Reimers and Iryna Gurevych. 2019b. [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.
- Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.
- Shivashankar Subramanian, Xudong Han, Timothy Baldwin, Trevor Cohn, and Lea Frermann. 2021. [Evaluating debiasing techniques for intersectional biases](#). *CoRR*, abs/2109.10441.
- Henry Weld, Xiaoqi Huang, Siqi Long, Josiah Poon, and Soyeon Caren Han. 2021. [A survey of joint intent detection and slot-filling models in natural language understanding](#). *CoRR*, abs/2101.08091.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick 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.
- Bianca Zadrozny. 2004. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114.

Cross-Encoder Data Annotation for Bi-Encoder Based Product Matching

Justin Chiu Keiji Shinzato

Rakuten Institute of Technology

Rakuten Group Inc.

{justin.chiu, keiji.shinzato}@rakuten.com

Abstract

Matching a seller listed item to an appropriate product is an important step for an e-commerce platform. With the recent advancement in deep learning, there are different encoder based approaches being proposed as solution. When textual data for two products are available, cross-encoder approaches encode them jointly while bi-encoder approaches encode them separately. Since cross-encoders are computationally heavy, approaches based on bi-encoders are a common practice for this challenge. In this paper, we propose cross-encoder data annotation; a technique to annotate or refine human annotated training data for bi-encoder models using a cross-encoder model. This technique enables us to build a robust model without annotation on newly collected training data or further improve model performance on annotated training data. We evaluate the cross-encoder data annotation on the product matching task using a real-world e-commerce dataset containing 104 million products. Experimental results show that the cross-encoder data annotation improves 4% absolute accuracy when no annotation for training data is available, and 2% absolute accuracy when annotation for training data is available.

1 Introduction

Product matching refers to the task of determining whether two different entries in the product catalog refer to the same real-world product. It is a core task for an e-commerce company where product catalog entries come from different sources and duplicates need to be identified and managed. There are two popular types of deep learning models that had been applied to the recent product matching work; cross-encoder (Li et al., 2020; Peeters et al., 2020) and bi-encoder (Shah et al., 2018; Tracz et al., 2020). Figure 1 depicts the architectures to demonstrate the difference between two models. When you have textual data for two products such as title pairs, the cross-encoder encodes them jointly, and

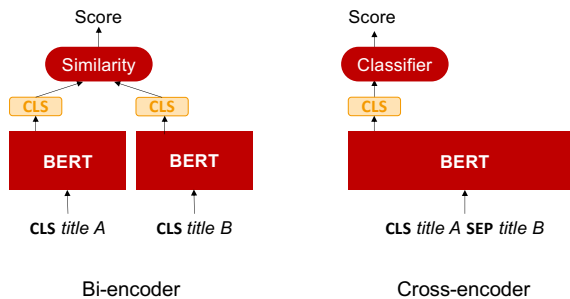


Figure 1: Bi- and cross-encoders using BERT.

the interaction between the two occurs through all encoder layers. The bi-encoder encodes both of them separately and there is no interaction between them until computing similarity.

The cross-encoder can capture more context due to its more complex interaction between the two inputs of data. However, because of its complexity, the model requires much more computational resources and time. A previous work (Reimers and Gurevych, 2019) reported that it takes 65 hours to find the most similar pair from a collection of 10,000 sentences using the cross-encoder (which requires us building 100,000,000 pairs for cross-encoder to process), while it only takes 5 seconds to encode all 10,000 sentences and compute cosine similarity for every possible pairs using the bi-encoder; that is 46,800 times difference in processing time. Since an e-commerce platform could easily have millions of products in its product catalog, such a volume will be a great challenge to use an approach that requires lots of computing resources, such as the cross-encoder. As a result, the bi-encoder is more viable for product matching in actual production systems.

In this paper, we propose cross-encoder data annotation, which is a technique that benefits from both model architectures. We first apply a cross-encoder on training data for a bi-encoder, and then train a bi-encoder using the data with high prediction scores from the cross-encoder. This process

is a way to utilize the knowledge learned in the cross-encoder to support the bi-encoder training. This approach is inspired by knowledge distillation (Hofstätter et al., 2020), where information learned by the cross-encoder is passed to the bi-encoder through the loss function. This can also be considered as a semi-supervised approach for product matching, where the new data is unlabeled.

In this paper, we make three major contributions. First, we demonstrate that our cross-encoder data annotation is effective for the product matching task on a real-world e-commerce dataset. Selecting the subset that both cross-encoder prediction and human annotation consider positive can train a better performance model in comparison to baseline. This can happen in an e-commerce company where different existing trained models are available for research purposes and new data are coming in. When building a model with the human annotated new data, our approach is applicable.

Second, we show that even when the human annotation of the bi-encoder training data is not available, we can use the prediction result from the cross-encoder as the data annotation, and build a model that performs better than a model based on product search results. This scenario can happen when reliable annotation for the new data is not available, and we can use the prediction of the cross-encoder to replace human annotation.

Finally, from the first contribution, we get to train a better model with less training data. This is not a common practice in deep learning since more training data is usually preferred. Our analysis shows that through the cross-encoder data annotation, we focus our training data on less queries. Since the bi-encoder tries to learn the difference between the two, the cross-encoder data annotation removes extreme samples in training data, which acts similarly to noise removal.

2 Product Matching

Given a product entry, a system is required to find entries in a product corpus that represent the same product, despite that the content in the entries are different. A product entry contains a set of information for a specific product, such as title, description, image, or categories. The multiple entries for the same product can be created by different vendors using the same e-commerce system. Assuming that our product corpus C contains M product entries for N different products $C = \{p_1, p_2, p_3, \dots, p_M\}$,

when given a query product p_q , we want to find a target product p_t that matches the query product in the corpus. The product corpus can be the product database for a deployed e-commerce system, which means the corpus size can easily contain over tens of millions of product entries.

We consider product matching as an information retrieval problem. We learn the similarity between the product entries, and use the similarity between query product p_q and target product p_t to decide whether two product entries are for the same product. This approach does not have to re-train the model every time the number of products N changes. By increasing the N in the product corpus C , we can increase the coverage of our product matching system.

3 Real-World E-Commerce Scenario

As a solution to the real-world product matching, there are two different types of approaches. The first is an approach based on a bi-encoder and another is based on a cross-encoder. We refer to the bi-encoder as *known bi-encoder*, and the cross-encoder as *known cross-encoder*. These known models require annotated data, which can be considered as the system that already exists in the e-commerce platform or systems that researchers develop as a proof-of-concept.

Each time a new item is received, sellers on the e-commerce platform upload the product data to the product database. Since the uploaded data sometimes contain incorrect information, a business unit in the e-commerce company periodically inspects if product data is correct. These inspection results can be regarded as human annotated data. Therefore, regardless of the annotation, new data is accumulated in the e-commerce platform. To incorporate the latest product information into a production system, we want to build a *new bi-encoder* using the new data. We exploit the known bi-encoder and known cross-encoder to train the new bi-encoder effectively, which is our key contribution for this work.

In the rest of this section, we describe the known bi-encoder, the cross-encoder, and how to train the new bi-encoder using these existing models.

3.1 Known Bi-Encoder

The known bi-encoder serves as a simple search engine to create training data for the known cross-encoder and new bi-encoder. We train the known

bi-encoder using triplet loss and in-batch negative, as reported in previous work (Karpukhin et al., 2020) for the open-domain Question Answering (QA) task. One key difference from the previous work is that we use one encoder to encode product title pairs although the bi-encoder for the QA task requires two different encoders, one to encode the question, and another to encode potential answer passages. We use BERT (Devlin et al., 2019) as an encoder, and regard the embeddings of [CLS] token as a representation of the given product title.

After training the bi-encoder, we encode our entire product corpus with the trained model, and then index them using FAISS (Johnson et al., 2019) offline. FAISS is an open-source library for similarity search that can be easily applied onto billions of vectors. After we have our entire corpus indexed, whenever we received a new query product, we can encode it with the same model and retrieve top k product titles from the FAISS index that are closest to the encoded query in the product embedding space.

3.2 Known Cross-Encoder

The known cross-encoder serves as a complex model you can built from the annotated training data to capture the most contextual information between product pairs. We employ BERT as an encoder, and put a classifier layer on the top of BERT. We convert product title pairs in the form of [CLS] Title 1 [SEP] Title 2. We regard the embeddings of [CLS] token obtained from BERT as a representation of the title pairs, and feed it to the classifier layer to judge if both titles refer the same product.

3.3 New Bi-Encoder

The new bi-encoder is the model we want to build when new data becomes available. The new data might come with reliable annotations or not, yet we still want to train model from it so our model can capture the most up to date information.

To train the new bi-encoder, we first retrieve relevant products for each query product using the known bi-encoder, and then apply the known cross-encoder to product pairs constructed from query and relevant products. For the pairs that the known cross-encoder predicts as matching pairs, we annotate positive pairs as training data. We call this approach *cross-encoder data annotation*. When using the cross-encoder data annotation, no human annotations for this training set are required, which

we believe will be a convenient scenario in a real-world, large-scale setup. Similarly to the known bi-encoder, after training the new bi-encoder, we index the entire product corpus using the model and FAISS.

When building the new bi-encoder, in the ideal scenario, annotation of the training data will be accurate and available so we can simply use the human annotations to decide what are the matching pairs. We can further improve the quality of human annotation by using the known cross-encoder to annotate the same data, and then use the intersection of both sets as positive training data. This will lead to a smaller training set than human annotation. However, there could be situations where such high quality human annotations are not available. As such, we directly use the set annotated by the known cross-encoder as positive training data.

4 Experiments

Our experiments are focused on the new bi-encoder. There are two main scenarios for our experiments, depending on whether the human annotations of the new products are available. For each scenario, we compare the experimental result with and without the cross-encoder data annotation. When the human annotations of the new products are available, we intersect human annotations with cross-encoder data annotations to improve the quality of training data. When the human annotations of the new products are not available, we use the cross-encoder data annotation for model training.

In future work, we will compare the new bi-encoder with a bi-encoder model trained with all query products used for training the known bi-encoder, known-cross encoder, and the new bi-encoder. The reason why we skip this comparison is that the goal of our experiments is to see how changing the annotation of the same data such as doing intersection with other annotations can improve models.

4.1 Dataset

Our experiments are based on our in-house dataset. The dataset contains product entries that are created by sellers on our e-commerce platform, and each entry consists of product ID,¹ title and description written in Japanese. The entries that refer the same product have the same product ID. The total number of products is 104 million. We regard this

¹More precisely, it is a global trade item number (GTIN).

Parameter	Cross-encoder	Bi-encoder
Batch size	128	128
Max seq. length	256	64
Learning rate	1e-05	1e-05
Temperature	n/a	1.0
Warmup rate	0.1	n/a
Vocabulary size	32,000	32,000
Max epoch	10	20

Table 1: Hyper-parameters for each model.

dataset as the product corpus. We only use the title in the product entry for model training. This avoids the mismatch where some sellers provide rich product descriptions while others provide limited or no descriptions.

4.2 Model

As encoders for cross- and bi-encoder models, we adopt BERT base (Devlin et al., 2019) in Japanese from Huggingface² and use its tokenizer to segment product titles into sub-words. We convert all characters in the titles into full-width before the segmentation. The average length of a product title is 27 sub-words.

The hyper parameters we used for training are reported in Table 1.

4.3 Training

To train the known bi-encoder, known cross-encoder, and new bi-encoder, we use three unique sets of 110K products as query products. We selected these query products from the results of business operation provided from a business unit in the company. Each query product has a product ID that the business unit assigned through the operation. There is no overlap of the product IDs within the three sets of the query products. We use 100K of the products as a training set and 10K products as a development set.

4.3.1 Known Bi-Encoder

For each entry in the first set of 110K query products, we randomly select a product title in the corpus having the same product ID, and we use these selected pairs as positive pairs for training. For negative pairs, we adopt the in-batch negative strategy proposed by Karpukhin et al. (2020).

²<https://huggingface.co/cl-tohoku/bert-base-japanese>

4.3.2 Known Cross-Encoder

The known cross-encoder is trained on the second set of 110K query product titles. We search these 110K product titles on the product corpus with the known bi-encoder to collect top 50 products for each query. This creates 5.5 million pairs of product titles, including both positive and negative pairs. Five million pairs are used as training data and the rest as development data. We randomly select 2.56 million pairs from the five million pairs and build the known cross-encoder. The benefit for constructing training pairs following this approach is to incorporate hard negative pairs into the training. Since hard negative pairs are similar in text, but do not refer to the same product, we can expect that the model learns the difference between products such as differences in product color.

4.3.3 New Bi-Encoder

The new bi-encoder is built on the third set of 110K query product titles and is the main focus of this paper. The set is similar to the situations where new products come into the product corpus. We want to build the bi-encoder using these new data. The new bi-encoder is built differently depending on whether the annotation for the new data is available.

When Available For each query product, we search the product corpus with its title using the known bi-encoder and collect the top 50 products. After collecting the top 50 products, we first use our human annotation to create training pairs based on all matching according to the annotation between the query product title and the retrieved product titles. This setup creates 113,374 training pairs that we called *human* annotated training pairs. We then use the known cross-encoder to predict all the product pairs in the top 50 retrieved results. For all the pairs that the known cross-encoder predicted as a match, we called them *cross-encoder* annotated training pairs, which have 107,059 pairs. Lastly, we perform an intersection on the human annotated training pairs and the cross-encoder annotated training pairs to utilize the knowledge in both pairs.

As a result, we obtain 84,688 training pairs which we called *intersection* pairs. The bi-encoder trained with the intersection pairs is our proposed approach for this scenario.

When Not Available The bi-encoder trained with the cross-encoder annotated training pairs is our proposed approach.

Training data	Accuracy
Human (baseline)	0.7356
Intersection (ours)	0.7575

Table 2: Results when new data annotation is available.

4.4 Baselines

We prepare different baselines depending on whether the annotation for the new data is available. When the annotation is available, the model trained with the human annotated pairs is a baseline. On the other hand, when the annotation is not available, we first search the query product titles on product corpus with the known bi-encoder. We still collect the top 50 retrieved results. However, since the annotation is not available, we only use all the top 1 retrieved results for every query product to create product pairs for training the baseline. Since we have one pair per query, we will have 100,000 training pairs in this setup. We call this dataset *top 1 product pairs*.

4.5 Evaluation

We select 9,991 products from the operation results, and use them as evaluation data. The data has less than 2% overlap between each of the 110K product sets described above. This set can help us understand whether our models can be effective to the product that is not in our training data.

For the evaluation, we chose a model with the lowest loss value on the development set. The evaluation measure is the top 1 accuracy of the search result. We check if the top 1 retrieved product and the query product are the same product.

4.6 Results

Tables 2 and 3 show the results for using our cross-encoder data annotation. When the annotation of training data is available, we can further refine the quality by conducting an intersection between the human annotation and the cross-encoder data annotation. Since the human annotated training pairs is larger than the intersection pairs, we can also observe that more training data does not guarantee better performances. When the annotation is not available, we can see that the model with the cross-encoder data annotation outperforms the baseline, which relies on the retrieval approach, to form positive training data. Our cross-encoder annotation result is also slightly better than the human annota-

Training data	Accuracy
Top 1 product pairs (baseline)	0.6985
Cross-encoder (ours)	0.7423

Table 3: Results when new data annotation is not available.

Training data	# of query products
Human	46,160
Cross-encoder	42,186
Intersection	36,872

Table 4: Number of query products in different training data.

tion result. This could be caused by the randomness on the training data, or our cross-encoder annotation is focused on lesser queries, which will be discussed in the section 5.1.

5 Analysis

Since our experiments use the identical modeling approach, we focus on understanding the composition of training data and how such differences in training data affects the prediction performance.

5.1 Analysis of Training Data

We studied how many matching pairs are created for each query. In the retrieval phase, since we use every matching pair available, it is possible to have multiple matching pairs for a single query product. Table 4 shows the number of query products in each set. Note that although we have 100K query products to search the corpus to form query pairs, we only have positive product pairs constructed from 46K product queries in the human annotation. This is because there are query products that the known bi-encoder does not return matching products in the top 50 results. Those query products cannot form any training pairs. As such, even if the human annotation training data have 113K pairs, they are from 46K distinct products. The number of query products dropped to 42K for the cross-encoder training pairs, and further lowered to 36K for the intersection of both sets. This reduced number of queries makes training process focus on the product pairs that are relatively easy to judge as the same product. In other words, removing some extreme training instances is effective to train better models, and it might work similar to removing data noise. A set

Title 1	Title 2
【1ケース】ファンタグレープ 160 ml 1缶	【送料無料】 コカ・コーラ ファンタグレープ 160 ml 1缶 30入 果汁ブレンドのフルーティーなおいしさ 果汁1%配合 【コカコーラからお客様へ直接お届けします】 【代引不可】
[1 case] Fanta Grape 160 ml Can	[Free Shipping] Coca-Cola Fanta Grape 160 ml Can 30 Packs Fruity Taste of Fruit Juice Blend Contain 1% Fruit Juice [From Coca-Cola to you directly] [Cash on delivery is not available]

Table 5: Example of a positive title pair removed by the intersection process (top) and its translation (bottom).

Type	# of pairs
Both incorrect	2,227
Only human correct	196
Only intersection correct	414
Both correct	7,154

Table 6: Number of title pairs in the test set for each correct/incorrect type.

of less diverse training data could model the most common difference between the two product titles and contribute to better performance.

Table 5 shows an example of a positive product title pair removed by the intersection process. Even though they refer to the same product, and the identical product name (*i.e.*, *Fanta Grape*) also shows up in both titles, we can observe the differences in both titles. In addition to the product name, the product title 2 contains extra descriptions such as a manufacturer name, a quantity, and shipping information. Learning from pairs with unbalanced product information might impact the effectiveness of features such as manufacturer names and quantities, which affects the overall performance of the trained model. We can expect that the intersection process removes such unbalanced pairs from the training data.

5.2 Analysis of Prediction Results

Table 6 shows the numbers of title pairs in the test set when we categorize the pairs according to the judgment results of the human and intersection models. From the table we can see that the number of the pairs in “Only intersection correct” is two times larger than that in “Only human correct.” To see what kind of product pairs in the test set the

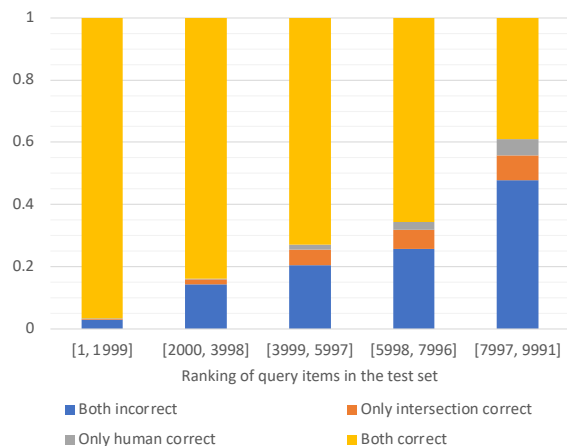


Figure 2: Ratio of correct and incorrect predictions of each model at each range. The x-axis shows the ranges of ranking when we sort title pairs in ascending order according to the distance returned from FAISS for the human annotation model. The title pair with shorter distance is at higher ranking.

cross-encoder data annotation effectively works, we investigate squared Euclidean (L2) distance between two product titles returned from FAISS for our human annotation model. We first sort the title pairs in ascending order according to the distances returned from the human annotation model, and then categorize the title pairs into five ranges equally. After that, for each pair in each range, we check if the prediction of the intersection model is correct.

Figure 2 shows the ratio of correct and incorrect predictions of each model at each range. From the figure, we can observe that the portion labeled “Only intersection correct” gets larger as the ranking gets lower, and is always larger than the portion of “Only human correct.” This means that our in-

tersection model has improvement on the subset of pairs that are far in distance in the embedding space created by the human annotation model.

5.3 Analysis for Continuous Improvement

As new data keep coming into the system in the real-world e-commerce scenario, we can continuously improve our models by integrating the training data of new model into the training data of known models. However, given the training data of updated cross-encoder model has more negative pairs comparing with positive pairs, this updated might have class imbalance issue where it will be mostly negative pairs. This can be addressed by setting a threshold during training data updates to ensure certain percentage of pairs that need to be positive pairs, and ensure the proper class balance in the updated training data.

6 Related Work

Product matching is a fundamental task for an e-commerce platform. Given the text in product entries, we want to match it to a specific product so duplicates can be identified. Earlier works tried to solve it by extracting defined product attributes and perform matching based on the extraction results (Mauge et al., 2012; Ghani et al., 2006).

Recently, more efforts have shifted toward focusing on text (Shah et al., 2018; Tracz et al., 2020). This avoids the need of doing attribute extraction and can directly be used on product titles and descriptions. There are two main directions for these efforts. One is considering the product matching task as an extreme classification problem, and another considering it as a zero-shot learning problem. As an extreme classification problem (Shah et al., 2018), the paper built a multi-class classifier that categorizes each input product information into a class, whereas each class represents a different product. The challenge is how to manage a multi-class classifier with several millions of classes, and the need to retrain the classifier every time a new product has entered the database. On the other hand, when considering a zero-shot learning problem, it focuses on learning the difference between the product texts (Tracz et al., 2020; Xiong et al., 2020). This makes it easier to apply the model on new products and more ideal in a production scaling environment. Both cross-encoder and bi-encoder had been used for solving product matching as a zero-shot learning problem. How-

ever, the computational complexity of the cross-encoder makes it hard to scale millions of items. For the bi-encoder models, different loss functions (Reimers and Gurevych, 2019; Tracz et al., 2020) had been applied to the task but the approaches are fundamentally similar.

In addition, there are several works in different domains that inspired our paper. Knowledge distillation (Hofstätter et al., 2020) is proposed to let a teacher model instruct a student model through learning. While their focus is on training, we applied similar ideas for data annotation. The bi-encoder approach for retrieval was also used in the open-domain QA task (Karpukhin et al., 2020; Yamada et al., 2021). Though we do not need two separate encoders for the question and answer separately, the retrieval task is still similar in implementation. There are also works (Luan et al., 2021) focused on analysis representation for text retrieval. Our cross-encoder and bi-encoder models also use the difference in the representation created by different encoder for the product matching task.

7 Conclusion

We demonstrated that we can use a cross-encoder to provide data annotation and to improve product matching performance on a bi-encoder. While such an approach can be useful when human annotation for new data is not available, it can also improve the quality of human-annotated data by conducting intersection. Our empirical analysis suggests that the intersection of our cross-encoder annotation and human annotation creates more focused training data that improves the quality of the product embedding space. As a result of this annotation technique, we obtained a new way to improve human annotation quality or building bi-encoder model without human annotation for product matching.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments.

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.
- Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. 2006. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48.
- Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*.
- Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. [Sparse, dense, and attentional representations for text retrieval](#). *Transactions of the Association for Computational Linguistics*, 9:329–345.
- Karin Mauge, Khash Rohanimanesh, and Jean-David Ruvini. 2012. [Structuring E-commerce inventory](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 805–814, Jeju Island, Korea. Association for Computational Linguistics.
- Ralph Peeters, Christian Bizer, and Goran Glavaš. 2020. Intermediate training of bert for product matching. *small*, 745(722):2–112.
- 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.
- Kashif Shah, Selcuk Kopru, and Jean-David Ruvini. 2018. [Neural network based extreme classification and similarity models for product matching](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 8–15, New Orleans - Louisiana. Association for Computational Linguistics.
- Janusz Tracz, Piotr Iwo Wójcik, Kalina Jasinska-Kobus, Riccardo Belluzzo, Robert Mroczkowski, and Ireneusz Gawlik. 2020. [BERT-based similarity learning for product matching](#). In *Proceedings of Workshop on Natural Language Processing in E-Commerce*, pages 66–75, Barcelona, Spain. Association for Computational Linguistics.
- Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.
- Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. [Efficient passage retrieval with hashing for open-domain question answering](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 979–986, Online. Association for Computational Linguistics.

Deploying a Retrieval based Response Model for Task Oriented Dialogues

Lahari Poddar *

Gyuri Szarvas *

Cheng Wang

Jorge Balazs

Pavel Danchenko

Patrick Ernst

Amazon

{poddarl, szarvasg, cwngam, jabalazs, danchenk, peernst}@amazon.com

Abstract

Task-oriented dialogue systems in industry settings need to have high conversational capability, be easily adaptable to changing situations and conform to business constraints. This paper describes a 3-step procedure to develop a conversational model that satisfies these criteria and can efficiently scale to rank a large set of response candidates. First, we provide a simple algorithm to semi-automatically create a high-coverage template set from historic conversations without any annotation. Second, we propose a neural architecture that encodes the dialogue context and applicable business constraints as profile features for ranking the next turn. Third, we describe a two-stage learning strategy with self-supervised training, followed by supervised fine-tuning on limited data collected through a human-in-the-loop platform. Finally, we describe offline experiments and present results of deploying our model with human-in-the-loop to converse with live customers online.

1 Introduction

A Task Oriented Dialogue (TOD) system aims to accomplish specific tasks such as hotel reservation (Budzianowski et al., 2018), flight booking, customer support (Moore et al., 2021) and so on. An end-to-end TOD system directly takes a multi-turn dialogue context as input and predicts the next response with a single model (Wen et al., 2016). These can be developed using either retrieval-based approaches (Tao et al., 2021; Chen et al., 2017) where the model ranks a response from a pre-constructed response pool; or generative approaches where a response is sequentially generated with encoder-decoder architectures (Serban et al., 2017; Sordoni et al., 2015). Although generative models are widely studied in literature for dialogue systems (Hosseini-Asl et al., 2020; Yang

*These authors contributed equally

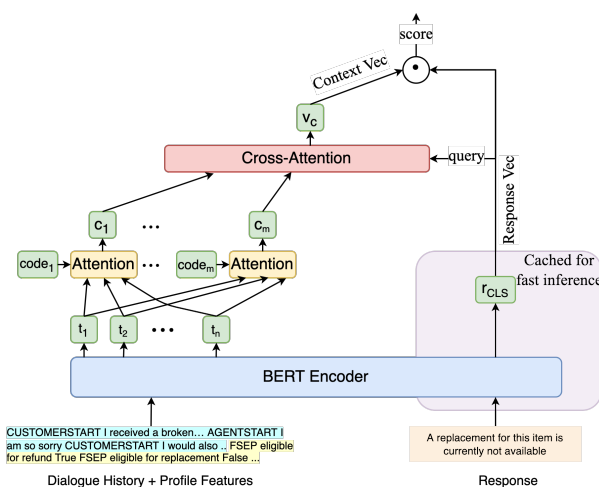
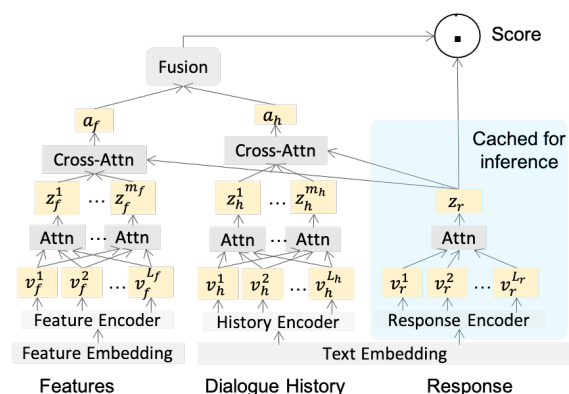


Figure 1: Production Ranking Models. The dialogue history, response and profile features are encoded with transformers (top) or using a shared Bert. Cross-attention layers learn the semantic correlation between history, features and candidate response. A score function computes and ranks candidate responses.

et al., 2021) as they are capable to generate free text, it is nearly impossible to provide guarantees on the style, quality and privacy risks for their real-world applications.

In this work, we focus on the development and

deployment of a retrieval-based conversational system for an online retail store, in the customer service domain.

Our main contributions are:

1. We design a simple yet effective algorithm for generating a large, representative response pool from un-annotated dialogues and show that it can achieve high coverage for handling natural language conversations.
2. We present an approach which combines self-supervised training (from human-human conversations) and supervised fine-tuning (from human-in-the-loop interactions) for learning dialogue models in real industry settings.
3. We enhance state-of-the-art Poly-Encoders architecture for retrieval based dialogue system, incorporating multi-modal information from dialogue text, and non-textual features associated with the order and the customer.
4. We present a breakdown of development and deployment stages of the conversational system from offline evaluation → controlled human-in-the-loop setting → fully online on live traffic with real customer contacts.

2 Related Work

Retrieval-based dialogue systems (Tao et al., 2021) involve single- and multi-turn response matching (Chen et al., 2017; Lu et al., 2019; Henderson et al., 2019; Gu et al., 2020; Whang et al., 2020; Poddar et al., 2022; Xu et al., 2021; Vig and Ramea, 2019). The selection of an appropriate response is usually based on computing and ranking the similarity between context and response. Two popular model architectures for such similarity computation between inputs, is Cross-encoders (Wolf et al., 2019), which perform full self-attention over a given input and label candidate; and Bi-encoders (Dinan et al., 2018), which encode the input and candidate separately and combine them at the end for a final representation. Bi-encoders have the ability to cache the encoded candidates, and reuse their representations for fast inference. Cross-encoders, on the other hand, often achieve higher accuracy but are prohibitively slow at test time. A recent method, Poly-encoders (Humeau et al., 2019), combines the strengths from the two architectures, and allows for caching response representations while

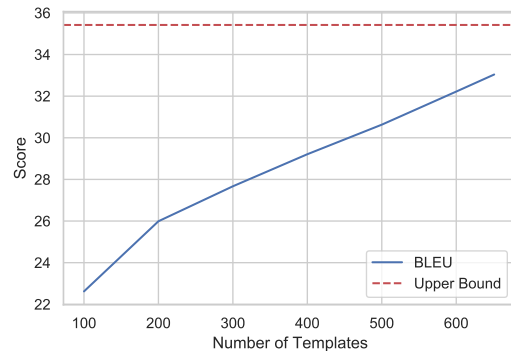


Figure 2: Template coverage on general conversations for Return Refund intent. Upper bound is established by adding templates to the pool based on human expert suggestions through several months of active use.

implementing an attention mechanism between context and response for improved performance. Transformer-based architectures (Vaswani et al., 2017; Devlin et al., 2019) are widely used to encode information in TOD systems. For instance, TOD-BERT (Wu et al., 2020) incorporates user and system tokens into the masked language modeling task and uses a contrastive objective function to simulate the response selection task. In this work, we also adapt the Transformer architectures and enhance Poly-Encoders to encode conversational history, response and profile features.

3 Response Pool Creation

We semi-automatically extract a broad template pool from a large number of anonymized human dialogues. We first select the template texts from human responses in actual dialogues. This ensures that the bot language conforms to the desired style.

Our primary selection criteria for response candidates are frequency and novelty. We iteratively select sentences that are (1) most frequently used in human dialogues, and (2) contain information different from already selected responses (detailed algorithm in Appendix A). This directly maximizes the dialogue model’s coverage, as measured by the fraction of contexts for which the model has a suitable response in the pool. An alternative approach would have been clustering frequent sentences and selecting a representative for each cluster (Hong et al., 2020) as templates. We instead opted for the deterministic procedure which is more intuitive for ingesting prior linguistic knowledge and provides interpretability.

Quantitative Evaluation of Coverage: Figure 2

shows the BLEU score by aligning the best matching templates to unconstrained human-human dialogues. As can be seen, with the growing size of the template pool, the BLEU score approaches this upper bound, proving that the proposed approach can achieve strong conversational capacity. Similar to (Swanson et al., 2019), we see that a set of 500 – 1k sentences can achieve good coverage of domain-specific conversations.

Template Decorations: We enhance the templates through attaching metadata, like calls to external APIs and constraints on profile features. For example, a template ‘I have issued a refund to your credit card’, will have an action that triggers an API call for issuing the refund. Through profile feature constraints we enforce consistency requirements on the dialogue, for example, filtering out the above template if an order is not eligible for refund. This establishes guarantees that the bot is always consistent with business policies.

4 Model Architecture

We represent a dialogue $D_i = \{a_1, u_1, \dots, a_n\}$ as a set of user (u_i) and agent (a_i) turns. While conversing with a user, agents look up information related to the particular order and item to determine applicable business policies and constraints. This may include item category, its delivery status, whether it was already refunded, among others; we encode these as categorical features.

We create multiple input-output tuples by splitting a complete dialogue transcript at each agent turn (e.g. at turn k). The model learns to predict the next agent response (a_k) given the dialogue history so far and the features that encode item level, customer level information and applicable policies. We flatten the history into a single sequence by concatenating all agent and user turns (x_h). We introduce two marker tokens [AGENTSTART] and [USERSTART] to mark the beginnings of an agent and user turn respectively. The features are also represented as a sequence, where each feature is encoded as a *key_value* pair (x_f).

Figure 1 presents the overall architecture of our proposed ranking model extended from Poly-Encoder (Humeau et al., 2019). We use separate transformers (Vaswani et al., 2017) as encoder

blocks (\mathcal{M}) for all inputs and encode them as:

$$\mathbf{v}_h = \mathcal{M}_h(x_h), \mathbf{v}_h \in \mathbb{R}^{L_h \times d} \quad (1)$$

$$\mathbf{v}_f = \mathcal{M}_f(x_f), \mathbf{v}_f \in \mathbb{R}^{L_f \times d} \quad (2)$$

$$\mathbf{v}_r = \mathcal{M}_r(x_r), \mathbf{v}_r \in \mathbb{R}^{L_r \times d} \quad (3)$$

where d is the dimension of the output vector, and L_h, L_f, L_r are maximum sequence lengths of history, features and responses respectively, and x_r is the target response.

Over the sequences we apply self-attentions to obtain latent representations. We represent the response using a single vector $\mathbf{z}_r \in \mathbb{R}^d$. For the multi-turn dialogue context and feature sequence we learn m_h and m_f representations, respectively, i.e. $\mathbf{z}_h \in \mathbb{R}^{m_h \times d}$ and $\mathbf{z}_f \in \mathbb{R}^{m_f \times d}$. We use 300 history representations (m_h) and 50 profile representations (m_f) in our experiments.

To learn history-response and feature-response correlations we apply cross-attention layers.

$$\mathbf{a}_h = Att_{cross}(K = \mathbf{z}_h, V = \mathbf{z}_h, Q = \mathbf{z}_r) \quad (4)$$

$$\mathbf{a}_f = Att_{cross}(K = \mathbf{z}_f, V = \mathbf{z}_f, Q = \mathbf{z}_r) \quad (5)$$

where K, V, Q present the key, value, query respectively, $\mathbf{a}_h \in \mathbb{R}^d$ and $\mathbf{a}_f \in \mathbb{R}^d$ are the final history and profile representations.

We then merge the two modalities of information from history and profile features through a 2-layer MLP to represent the complete dialogue context:

$$\mathbf{a}_{hf} = \mathcal{F}([\mathbf{a}_h, \mathbf{a}_f]) \quad (6)$$

A score function is used to rank the candidate responses given a (history, profile) pair through computing similarity using dot-product.

$$\mathbf{s} = f_{score}(\mathbf{a}_{hf}, \mathbf{z}_r) \quad (7)$$

We train the model in end-to-end manner using a binary cross-entropy loss.

5 Model Development

In order to protect customer experience and trust, we do not simply train a model on human-human conversation data and deploy it to live traffic directly. To utilize the expertise of our customer service agents, we introduce a subsequent stage that not only acts as an intermediate test-bed, but also provides a fly-wheel to annotate data. Our model training consists of the following two stages.

5.1 Self-supervised Training

A large volume of anonymized human-human conversations is used for learning an initial dialogue model via Self-Supervised Training (SST). The goal is to rank the correct next utterance higher compared to other randomly sampled utterances given the dialogue history and associated profile features. Note that this model is independent of the response pool discussed in Section 3.

5.2 Supervised Fine-Tuning

We use the model obtained from the previous stage to collect supervision data within a human-in-the-loop environment. In this setting, whenever a customer starts a contact, the utterance, along with profile features, is passed on to the SST model. The entire response pool is ranked by the model and top k responses are shown to human agents. They have three options for responding to the customer- a) accept the suggestion, b) pick a different template from the pool, c) indicate a failure of the pool (no response in the pool can be used to progress the conversation) or the constraints (e.g. a refund should be offered but it is not available).

We utilize the data collected through this human-in-the-loop setup for further Supervised Fine-Tuning (SFT) of the model. The key difference compared to the previous stage is that in this setup both the positive and negative responses come from the response pool. We create a set of N candidates with the one that the human expert accepted or searched as positive. The negative candidates are sampled randomly from the template pool. Whenever the response used by the expert was obtained through search, we leverage the model-suggested responses as hard negatives.

The primary goal of this human-in-the-loop stage is to collect the best possible data for supervised training. Instead of the straightforward approach of suggesting the top-scored template to human experts, we found that a sampling strategy among high scoring templates can boost impressions for less frequent templates. This helps improving the utility of the collected data.¹

6 Evaluation

We conduct offline and online experiments on internal conversational datasets of an e-commerce customer service from the following two intents,

¹For space constraints, the implementation details and experimental results are found in Appendix C

	Start Return		Return Refund	
	Unlabeled	Labeled	Unlabeled	Labeled
# Dialogues	824K	30K	918K	21.6K
Avg. # Turns	18.9	13.5	30.6	8.7

Table 1: Overview of datasets.

1. **Start-Return (SR)**: where a customer wants to initiate a return of an item.
2. **Return-Refund Status (RRS)**: all post-return cases where customer may enquire about the status of a return or refund already issued / currently under processing.

6.1 Experimental Setup

Datasets. The dataset statistics are summarized in Table 1. We tokenize and sentence split dialogue turns using NLTK toolkit (Loper and Bird, 2002). We split each dataset to train/dev/test sets with ratio 90:5:5 and use the most frequent 30K, 10K tokens as dialogue encoder vocabulary for Start Return and Return Refund intent, respectively.

Model Training. We train and fine-tune the models on the unlabeled and labeled intent datasets, respectively. We use a learning rate of 0.00015 and train for 30 epochs with early stopping.

Metrics. For offline evaluation, we use the standard ranking metrics Recall@ k , and MRR (Mean Reciprocal Rank), and a metric for offline manual evaluation for top scored template, namely,

1. *Template Precision (TP)*: For 200 samples drawn randomly from test set, we use the model to rank templates in the pool. We manually evaluate the acceptability of the top-ranked template by the model and report an averaged precision.

For online evaluation we introduce:

2. *Turn-level Acceptance Rate (TAR@ k)*: $\frac{N_{accept}}{N_{total}}$, N_{accept} is the number of turns accepted by human expert out of the number of total turns N_{total} . TAR is an online correspondent of the *Recall@ k* metric. A higher value of TAR indicates model’s capability of handling a conversation well, through ranking of the template pool.
3. *Task Completion (TC)*: The percentage of contacts that agents were able to resolve - either by accepting model suggestion or searching the pool. TC measures the quality and capacity of the pool and sets an upper bound for the bot’s success rate.
4. *Automated Task Completion (ATC)*: Success rate of the deployed bot; i.e. the percentage of contacts where the system is able to resolve the customer issue, such that the same customer doesn’t

Intent	Offline Metrics			Online Metrics		
	Rec@1/29	MRR	TP	TAR@4	TAR@1	TC
SR	76%	86%	76%	–	71%	52%
RRS	71%	81%	71%	50%	17% [†]	39%

Table 2: Offline and Online results for initial dialogue model trained with self-supervision. [†] RRS was launched in top-4 suggestion mode, while the better performing SR intent was launched in top-1 suggestion mode.

repeat the contact within next 24 hours.

6.2 Self-Supervised Training Results

We first report offline and online results of models trained using human-human dialogues in self-supervised manner. We deploy the trained model in online human-in-the-loop setup (described in Section 5.2) and measure TAR@k and TC. We share our key learnings in this section.

Performance varies depending on domain complexity: From the results in Table 2, we first observe the significant gap in online metrics between the two intents, which shows that the performance of dialogue systems in real-world conversations are highly dependent on the complexities of the domain. This primarily reflects in the lower task completion rates (TC) of RRS, where dialogues often become open-ended when discussing issues with a previous return, compared to the more procedural dialogues about starting a return.

Human choices are often arbitrary among close alternatives: In the RRS intent we conducted the online experiment by displaying top-4 templates to the agents instead of a single one. This decreases TAR@1 by ~15%. Using more suggestions generally improves the agent’s productivity due to limited search. However, we observed that human choices among similar templates are often arbitrary, leading to performance drops.

Features are indisposable: For RRS the TAR@4 was also quite low, especially given that SR had TAR@1 above 70%. The main reason was the lack of crucial features, like granular tracking information from the carrier companies about the return package. This limited its ability to accurately condition on external factors compared to human experts. This underlines the saliency of features (or external knowledge) in a practical TOD setting.

Data-driven templates enable transfer learning: From the offline results we note that the manually annotated TP metric closely resembles Recall@1 for both intents. This implies that the model is able to learn from human-human conver-

Intent	Offline Metrics		Online Metrics	
	Rec@1/29	MRR	TAR@1	TC
SR	80.9%	89.1%	84.9%	56.6%
RRS	76.4%	84.9%	46.6%	46.3%

Table 3: Offline and Online results post finetuning

sations and apply it for ranking the restricted template set. Having a large template pool that follows a similar data distribution as the original agent responses helps in achieving this smooth transition. **Large template pool is effective for handling conversations at scale:** The contact-level metric (TC) shows that with the generated template pool 52.1% contacts could be fully resolved for the SR intent. This demonstrates the potential of using a large representative set of agent responses for tackling in-domain task oriented conversations.

6.3 Results After Supervised Fine-Tuning

We explore two fine-tuning strategies with the limited data collected from human-in-the-loop stage. **Catastrophic Forgetting with training only on restricted language:** Figure 3a shows that fine-tuning with only the limited supervised data leads to better performance on the supervised test set (SFT) but increasingly worse performance on the general conversation test set (SST) as training progresses. This implies that as the model is being trained on this restricted data distribution, it is ‘forgetting’ previously learned knowledge through self-supervision. To mitigate this, we adopt a simple replay mechanism (Rolnick et al., 2019). We augment the fine-tuning dataset by mixing in equal number of training instances from the self-supervised dataset. As seen from Figure 3b, training with the balanced dataset leads to consistently better results on both datasets. This proves the ability of the model to learn from the limited supervised dataset without overriding previous knowledge. Similar results were observed for RR intent Figure 3b ((c)-(d))

Supervision from human-in-the-loop significantly boosts performance: Table 3 shows the performance after supervised fine-tuning. In this experiment, the offline test set contains template responses from the human-in-the-loop setup, in contrast to the general conversation responses considered in offline evaluation in Table 2. Offline metrics for both intents are generally higher compared to the general test set (Table 2). This is expected, since by restricting to specific template set

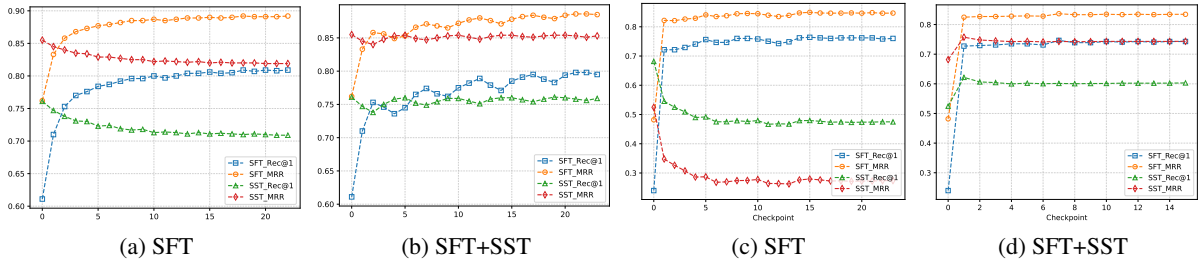


Figure 3: Evaluation after fine-tuning with labels collected from human-in-the-loop platform on the SR dataset ((a) and (b)) and the RR dataset ((c) and (d)). X-axis: checkpoints, Y-axis: performance.

the language variation in agent turns is greatly reduced, making the ranking task easier. More importantly, we observe the increase in the online metric TAR post fine-tuning, demonstrating the effectiveness of the two-stage training strategy.

Human-in-the-loop setup also augments the template pool: The increase in TC is independent of training strategies and fueled by enhancements to the template pool using suggestions from the human experts engaged in the supervised data collection process. It is noteworthy that the initial (automatically collected) template pool attained a high 92% and 84% of the TC compared to these refinements. This demonstrates the efficacy of the proposed template creation method (Section 3).

6.4 Deployment

A key advantage of the proposed model architecture is its inference efficiency. The textual representation of templates can be encoded once at initialization and cached for future calls. For each template only the cross-attention layers and final scoring needs to be re-computed. This lets the inference latency scale linearly with template set size (Figure 4). On `c5.2xlarge` CPU instances with $1k$ templates the latency is below 0.5 sec, which is sufficient for real time conversation with users; on a small GPU instance `g4dn.xlarge` up to $5k$ templates can be scored within 50 ms.

While the results after supervised fine tuning for SR reach sufficiently high quality for deployment of the chatbot, additional improvements are needed for RRS, especially in providing the essential package tracking information through features. The Start Return chatbot achieved 48.3% Automated Task Completion (ATC) after deployment.

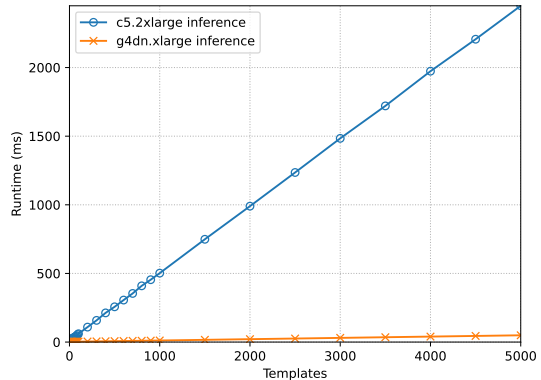


Figure 4: Inference speed with growing template pool

7 Developing with Pre-trained Language Models

We continue developing and improving our models after deployment. With the initial model launched, we explore plugging in pre-trained language models like BERT as our text encoder (Figure 1b). We adopt a single shared encoder to simplify our architecture and limit the memory footprint of training and hosting a large model. Additionally, we convert the item features to have more semantic names (e.g. ‘eligible for refund’) and append them to dialogue history for generating the complete context. This allows the self-attention mechanism of the transformer module to capture multi-modal interactions between features and dialogue turns that are grounded on them. We use batch negatives during training and learn the next response prediction task using categorical cross-entropy loss.

Table 4 shows the offline evaluation results (analogous to Table 2) for the two intents. Using PLMs clearly help in improving performance at a much higher data efficiency - with only 20% of training data the BERT initialized model out-

Data Size	Model	SR		RRS	
		Rec@1	MRR	Rec@1	MRR
100%	no PLM	76.1%	85.5%	71.2%	81.0%
20%	bert-base	83.2%	89.5%	76.3%	85.1%
50%	bert-base	88.2%	92.9%	81.4%	88.7%

Table 4: Offline performance comparison for pre-trained language model as text encoder

performs our previous model, which did not use any pre-training but was trained on 100% available in-domain dataset. We further fine-tune the models using the supervised data collected through the human-in-the-loop setup described in Section 5.2.

Next, we deployed the improved BERT model for SR intent to live customer traffic. Similar to the offline results, we observed a significant improvement in ATC to 55.3% for the BERT model compared to 48.3% for our previous model that did not use any pre-trained language model. In future we plan to deploy for the RRS intent as well and compare both performance and efficiency.

8 Conclusion

We presented a neural, retrieval-based dialogue model that ranks responses from a large, data-driven template pool. Pre-defined responses make it possible to enforce requirements for consistency to business policies and the proposed template mining method provides good conversational capacity. The model is accurate and efficient in terms of inference speed to handle conversations in real time. A human-in-the-loop setup lets us effectively collect a small-sized labeled dataset to improve the quality for online deployments.

Offline and online results demonstrate that this is a viable approach for developing TOD systems for practical usecases. While RRS showed good improvements with our training protocol, it needs further work to be deployed. Performance on the SR intent permitted the deployment of the model; its live success rate almost reaches the 56% upper bound that humans achieve in the controlled setting. Anecdotal evidence² from customer feedback shows that successful dialogues by the model provide good conversational experience.

²We include few positive user feedback in Appendix B.

Ethical considerations

Development and experiments. We used anonymized text dialogue snippets to train the models. The system predicts template responses, hence the model described in this work has no way to reveal customer information. This is actually a key theoretical advantage to generative models. We do not release the datasets used in the experiments.

Failure modes. Regarding risks related to system errors, incorrect predictions of the models described in this work may result in a confusing dialogue experience for customers. However, the practical risk related to such confusion is limited, because the chatbot operates in a semi-automation setting where it naturally predicts and transfers the contact to a human expert upon a drifting dialogue history. Moreover, customers also have an option to talk to a human associate upon request, if they consider the system doesn't work as expected.

References

- Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. [MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.
- Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186.
- Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations*.
- Jia-Chen Gu, Tianda Li, Quan Liu, Xiao-Dan Zhu, Zhenhua Ling, Zhiming Su, and Si Wei. 2020. Speaker-aware bert for multi-turn response selection in retrieval-based chatbots. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.
- Matthew Henderson, Ivan Vulic, D. Gerz, I. Casanueva, Paweł Budzianowski, Sam Coope, Georgios P. Spithourakis, Tsung-Hsien Wen, N. Mrksic, and Pei

- hao Su. 2019. Training neural response selection for task-oriented dialogue systems. In *ACL*.
- Teakgyu Hong, Oh-Woog Kwon, and Young-Kil Kim. 2020. End-to-end task-oriented dialog system through template slot value generation. In *INTER-SPEECH*.
- Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. [A simple language model for task-oriented dialogue](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.
- Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. 2019. Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring. In *International Conference on Learning Representations*.
- Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR'17*.
- Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70.
- Y. Lu, Manisha Srivastava, Jared Kramer, Heba Elfardy, Andrea Kahn, Song Wang, and Vikas Bhardwaj. 2019. Goal-oriented end-to-end conversational models with profile features in a real-world setting. In *NAACL*.
- Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *ICLR'17*.
- Kristen Moore, Shenjun Zhong, Zhen He, Torsten Rudolf, Nils Fisher, Brandon Victor, and Neha Jindal. 2021. [A comprehensive solution to retrieval-based chatbot construction](#).
- Lahari Poddar, Peiyao Wang, and Julia Reinspach. 2022. [DialAug: Mixing up dialogue contexts in contrastive learning for robust conversational modeling](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 441–450. International Committee on Computational Linguistics.
- David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. 2019. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32:350–360.
- Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.
- Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *NAACL-HIT*.
- Kyle Swanson, Lili Yu, Christopher Fox, Jeremy Wohlwend, and Tao Lei. 2019. [Building a production model for retrieval-based chatbots](#). In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 32–41, Florence, Italy. Association for Computational Linguistics.
- Chongyang Tao, Jiazhan Feng, Rui Yan, Wei Wu, and Daxin Jiang. 2021. [A survey on response selection for retrieval-based dialogues](#). In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4619–4626. International Joint Conferences on Artificial Intelligence Organization. Survey Track.
- 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.
- Jesse Vig and Kalai Ramea. 2019. Comparison of transfer-learning approaches for response selection in multi-turn conversations. In *Workshop on DSTC7*.
- Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.
- Taesun Whang, Dongyub Lee, Chanhee Lee, Kisu Yang, Dongsuk Oh, and Heuseok Lim. 2020. An effective domain adaptive post-training method for bert in response selection. In *INTERSPEECH*.
- Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. [Transfertransfo: A transfer learning approach for neural network based conversational agents](#). *CoRR*, abs/1901.08149.
- Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. 2020. [TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue](#). pages 917–929.
- Ruijian Xu, Chongyang Tao, Daxin Jiang, Xueliang Zhao, Dongyan Zhao, and Rui Yan. 2021. Learning an effective context-response matching model with self-supervised tasks for retrieval-based dialogues. In *AAAI*.
- Yunyi Yang, Yunhao Li, and Xiaojun Quan. 2021. [Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14230–14238.

A Response Pool Creation

a-to-z	book	confirmation	dropoff	hold	name	price	reimbursement	security	transaction
accept	box	contact	e-mail	id	notification	print	reorder	sell	transfer
access	business	correspondence	elaborate	ignore	number	priority	repeat	seller	transit
account	call	cost	email	inconvenience	option	problem	replace	sender	understand
action	cancel	create	error	inform	order	proceed	replacement	service	understanding
address	card	credit	escalate	information	pack	process	representative	ship	update
allow	carrier	cvc	exception	initiate	package	product	request	shipment	ups
alternative	center	damage	exchange	inventory	packaging	promo	require	shipping	url
amount	certificate	delay	expedite	investigate	party	promotion	research	solution	use
apologize	charge	deliver	experience	investigation	patience	provide	resolution	specialist	verify
apology	check	delivery	expire	issue	pay	purchase	resolve	status	visa
apply	checking	department	extend	item	payment	qr	responsibility	stay	wait
arrange	claim	detail	fee	label	perfect	quantity	restock	stock	waive
arrive	click	device	feedback	leadership	phone	re-order	restocking	store	warehouse
assistance	code	digit	find	link	photo	reason	resubmit	subscription	warranty
associate	come	disarm	follow	locker	pick	receipt	retrocharge	suggest	website
authorization	compensation	discount	fulfil	mail	pickup	receive	return	supervisor	window
availability	complaint	display	fulfillment	mailing	picture	refer	returnable	support	
balance	complete	dispose	fund	manufacturer	place	reflect	review	team	
bank	concern	disregard	gift	member	policy	refund	safety	time	
billing	condition	donate	guarantee	method	post	regard	scan	track	
birth	confirm	drop	help	money	prefer	register	screenshot	tracking	

Table 5: Important lemmas utilized for template selection.

Algorithm 1 Response Pool Generation Process

- 1: Preprocess data by sentence splitting, tokenization, part-of-speech tagging lemmatization.
- 2: Transform each sentence into a sequence of `verb`, `noun`, `adjective`, `adverb` lemmas by dropping punctuation and non-content words of other parts of speech.
- 3: Manually review top 1k frequent `verb` and `noun` lemmas to retain a list of keywords kw . ▷ We kept altogether 215 lemmas that can be found in Table 5, with ~30 minutes of manual effort.
- 4: Template set $T = \emptyset$
- 5: **for** sentence $s \in$ dataset **do** [in decreasing order of frequency]
- 6: **for** sentence $t \in T$ **do**
- 7: $sim(s, t) = exp(\frac{\sum_{k=1}^2 ln(J_k(s_n, t_n))}{2})$
- 8: $sim(s, T) = argmax_{t \in T}(sim(s, t))$
- 9: **if** $sim(s, T) < \lambda$ **then**
- 10: **if** $freq(s) > f_1$ **then**
- 11: $T = T + \{s\}$
- 12: **if** $freq(s) > f_2$ & $s \cap kw \neq \emptyset$ **then**
- 13: $T = T + \{s\}$
- 14: Manually remove sentences from T that have grammatical errors or are inappropriate for usecase (e.g. greetings). ▷ We used $\lambda = 0.4$, $f_1 = 350$, $f_2 = 15$. J_k denotes Jaccard similarity of unigrams ($k = 1$) and bigrams ($k = 2$)

B User Feedback

CHATBOT:	You can also leave a comment about how your experience went. This helps me improve.
USER:	Thanks so much!!! I was afraid I'd not get a refund, let alone get a return. Thanks so very much
CHATBOT:	Again, I am sorry for the trouble that you had faced due to this circumstance and but for now do you have some clarification or further question regarding with my resolution?
USER:	No thank you
CHATBOT:	How does that sound?
USER:	Sounds ok.....is there another product you would recommend that will work?
CHATBOT:	Would there be anything else I can do to help?
USER:	no. Thanks for your help Mr. or Mrs. Bot

Table 6: Examples for positive user feedback.

C Template Exploration

One of limitations that we observed from model online deployment is that our deterministic models always try to rank the high volume templates. This gives less opportunity to the templates with lower frequency. For instance, for some templates with similar semantic meaning, “*No worries, let me see what I can do to help you out*” is a high frequency template, while “*No worries, I will check that for you real quick.*” is a low frequency template. To improve the diversity of templates, we enable the deterministic ranking models with exploration capability by using Gumbel-Softmax Trick (Jang et al., 2017; Maddison et al., 2017), which is originally proposed to make discrete variables to be differentiable. Here we only use the sampling functionality with temperature to control the degree of exploration. The main idea is to replace the original sigmoid score function with Gumbel-Softmax. For each inference run, we sample a template based on the computed scores.

C.1 Dataset and Implementations

We collected the human-in-the-loop data from deployed deterministic ranking model and exploration model. Over around 8 weeks, we collected 67,136 samples as training set, 5000 and 8196 samples as validation and test set respectively, for each model. To make a fair comparison, we ensure the evaluated sets for each model are same. We have totally three types of datasets for evaluation: (1) \mathcal{D}_A : the test set is a combination of the test set of deterministic SFT data, exploration SFT dataset and general conversation SST dataset. (2) \mathcal{D}_B : SST test dataset; (3) \mathcal{D}_C : SST validation dataset.

We use the original ranking setting for experiment as described in Sec.3, and set temperature=1. We want to examine: (1) how fast the exploration model can help to explore and lift those tail templates; (2) and the predictive performance of exploration model as compared to deterministic model.

C.2 Exploration Results

Table 7 presents the accuracy, Recall@1 and MRR performance for each model. As we can see, exploration ranking model doesn’t hurt the original predictive performance when performing exploration on templates. This is important in real-world setting, because the degraded model perfor-

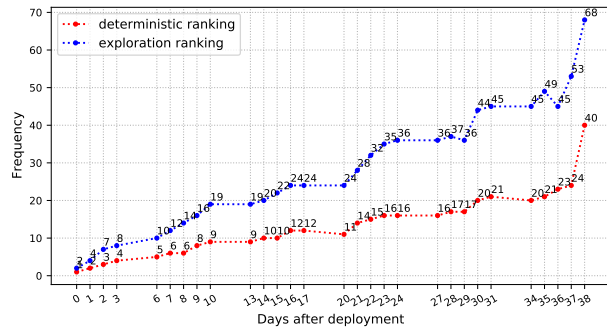


Figure 5: The median of cumulative frequency of templates over time (days) for deterministic and exploration models.

Model / Dataset	Loss	Acc	Recall@1	MRR
$M_d(\mathcal{D}_A)$	0.2186	0.9203	0.9430	0.9707
$M_e(\mathcal{D}_A)$	0.2161	0.9197	0.9434	0.9710
$M_d(\mathcal{D}_B)$	0.2284	0.9161	0.9397	0.9691
$M_e(\mathcal{D}_B)$	0.2257	0.9163	0.9397	0.9691
$M_d(\mathcal{D}_C)$	0.1756	0.9322	0.6094	0.7547
$M_e(\mathcal{D}_C)$	0.1749	0.9320	0.6162	0.7584

Table 7: The performance comparison between deterministic ranking model (M_d) and exploration ranking model (M_e) on different evaluated datasets.

mance usually leads to unsatisfactory customer experience.

Figure 5 demonstrates that the exploration ranking model helps to generate a target K impression for the average template 2-3 faster than that of deterministic ranking model.

Tackling Temporal Questions in Natural Language Interface to Databases

Ngoc Phuoc An Vo, Irene Manotas, Octavian Popescu,
Hangu Yeo, Elahe Khorasani, Vadim Sheinin

IBM Research

{ngoc.phuoc.an.vo, irene.manotas}@ibm.com
{o.popescu, hangu, elkh, vadims}@us.ibm.com

Abstract

Temporal aspect is one of the most challenging areas in Natural Language Interface to Databases (NLIDB). This paper addresses and examines how temporal questions being studied and supported by the research community at both levels: popular annotated dataset (e.g. Spider) and recent advanced models. We present a new dataset with accompanied databases supporting temporal questions in NLIDB. We experiment with two SOTA models (Picard and ValueNet) to investigate how our new dataset helps these models learn and improve performance in temporal aspect.

1 Introduction

Natural language interface to databases (NLIDB) is a task to bridge the gap between storing complex structured data in databases (DBs) and retrieving structured data from databases using natural language questions. An NLIDB system allows users to access information stored in a DB by using questions in natural language (e.g. English). In reality, temporal aspect is a common dimension existing in any DBs since it is realistic and practical to store data with associated date/time in DBs. However, it is challenging to retrieve information from DBs with temporal aspect due to difficulties and limitations such as 1) complex and limited association between certain entities/attributes and temporal info in DB that reflects different states of data in different time frames, and 2) complexity of natural language expressions to decipher the actual value of temporal expressions in given questions to correctly derive data from DBs. Temporal aspect is a common yet distinct and complex dimension that needs to be handled carefully in NLIDB.

Given its difficulties and complexity, temporal aspect in NLIDB is currently under-attended by the research community. First of all, it is difficult to design and create database that supports temporal dimension. Next, it is more challenging

to create pairs of questions and associated SQL queries that support the learning of temporal aspect in NLIDB (discussion in Section 4). Our contribution is threefold: i) we investigated how temporal aspect was defined and annotated in the well-known Spider dataset, ii) we created the new dataset TempQ4NLIDB supporting the learning and understanding temporal questions in NLIDB, and iii) we experimented with two SOTA models - ValueNet (Brunner and Stockinger, 2021) and Picard (Scholak et al., 2021) - to understand how they learn and handle temporal questions. To the best of our knowledge, this work is one of the first attempts trying to explore this research area to support temporal questions in NLIDB.

2 Background and Related Works

NLIDB or Text-to-SQL is a long standing NLP task with many references on its complexity and achievements obtained recently (Navid et al., 2017; Popescu et al.; Yao et al., 2010). Although there is no official definition of temporal questions in NLIDB, the study (Androustopoulos et al., 1995) mentioned about temporal questions with respect to temporal databases (Jensen and Snodgrass, 2018). In contrast, the study of temporal questions is a strong interest in Question Answering (QA) community. In the scope of this paper, we adopt a definition of temporal questions in QA for our work that is "A temporal question is any question, which contains a temporal expression, a temporal signal, or whose answer is of temporal nature." (Jia et al., 2018). In QA, temporal questions can be answered by temporal information embedded in semantic relations and timeline between events in corpus or taxonomy, whereas DB records associated with temporal dimension are answers in NLIDB.

The Spider dataset (Yu et al., 2018) and Spider challenge was introduced in 2018 where participating teams can have their systems evaluated on an unseen test set which is not available to public.

There are two Spider sub-challenges, the first one for SQL inference without values and the second for systems that handle the values in SQL queries. Many attempts with different approaches, based on neural networks, especially on encoder-decoder architecture, have continuously improved the state of the art (SOTA). First, the sequence-to-sequence approach was introduced (Cai et al., 2018; Gehring et al., 2017; Yin et al., 2016; Rabinovich et al., 2017) in which a neural network with very good proven qualities in translation tasks translates an English query into an SQL query. These systems infer the SQL formula directly as a standard translating task from one language to another.

Instead of translating into SQL, another approach translates questions into a representation that captures semantics of a question, namely Intermediate Representation (IR) (Guo et al., 2019; Zhang et al., 2019; Bogin et al., 2019). The network learns a more structured and compact form of the query itself. From IR to SQL is a deterministic process: a context free grammar is used to convert one into another. The authors build on the work of (Sun et al., 2019; Cheng et al., 2019) that used Abstract Syntax Tree (AST). The decoder infers the IR as an AST representation of the query. Among others, ValueNet (Brunner and Stockinger, 2021) not only uses IR approach but also extends the context free grammar to include values and became one of the best systems in the Spider challenge in 2020. Recently, PICARD (Scholak et al., 2021), a state-of-the-art algorithm for constrained decoding was introduced and achieved top rank in Spider challenge. It relies on the structure and the content of the database as well as the knowledge encapsulated in the T5 language model (Raffel et al., 2020).

3 Discovering Temporal Aspect in Spider

Spider (Yu et al., 2018) is a popular large-scale complex and cross-domain dataset supporting Text-to-SQL task consisting of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. Since there is no description of temporal aspect in Spider, we investigate temporal questions in this dataset.

3.1 Questions Related to Temporal Aspect

Questions related to temporal aspect are questions that query or process one or more column representing date/month/year/time in SQL queries.

Searching in Spider, among 7,000 questions in Train set, we found 504 SQL queries (7.2%) and 67 SQL queries (6.46%) among 1,036 questions in Dev set that hit our search keywords, respectively. Questions associated with these SQL queries are considered having temporal aspect.

3.2 Temporal Question Types

Among questions found in Section 3.1, we define three question types regarding temporal aspect:

Type 1: Questions querying Temporal Info.

This type of question only queries for temporal information from DBs using the SELECT clause and has no logical operation with temporal columns in database, such as: {What are the first names and **birth dates** of players from the USA?}

Type 2: Questions querying Temporal Information with Grouping or Sorting.

This type of question may or may not query temporal information but it has *GROUP BY* or *ORDER BY* processing on one or more temporal columns in database. These questions usually have one or more temporal adverbs like {*most recently, in the order of, youngest, oldest, longest, the latest*}. For examples: {What is the first name and country code of the **oldest** player?} or {How many total tours were there for **each ranking date**?}.

Type 3: Questions with Temporal Conditions.

This type of question may (not) query temporal information but always has one or more temporal conditions to derive required information from DB, such as: {Show the organizer and name for churches that **opened between 1830 and 1840**}. In this paper, we only focus on this type since it is the most practical and challenging type in reality. We found 201 and 19 temporal condition questions in Train/Dev sets of Spider, respectively.

4 TempQ4NLIDB Dataset for NLIDB

Since temporal dimension is practical and crucial for DBs in any real-world applications, and the lack of dataset for temporal questions in NLIDB, we created TempQ4NLIDB - a dataset of temporal condition questions for studying and experimenting with the recent advanced NLIDB models.

4.1 Accompanied Databases

We release two synthetic DBs (SQLite compatible) adopted from our real-world industry projects

	WH	HR
Temporal Train	65	46
Temporal Dev	7	4
Temporal Test	28	25
Total	100	75
Non-Temporal Train	146	99
Non-Temporal Dev	16	10
Non-Temporal Test	40	78
Total	202	187

Table 1: Temporal and Non-Temporal Questions in TempQ4NLIDB

and fully anonymized for public use. All temporal values are in standard format YYYY-MM-DD.

Human Resource (HR) is a one-table DB containing employees information such as employee number, name, birthdate, hire date, leave date, department, manager, salary, bonus.

Warehouse (WH) is a complex schema for sales activities consisting of 8 tables. More details can be found in our dataset.

4.2 Temporal Condition Questions and SQLs

We created new temporal conditional questions and SQL queries for two accompanied DBs. We also release the non-temporal questions and SQL queries for performance evaluation and comparison (Table 1). Our dataset is available here¹. We may continue expanding this dataset with more questions of different types and complexity levels in near future.

4.2.1 An ad-hoc Date Annotator (DA)

We use an ad-hoc date annotator (a part of another rule-based NLIDB system) (Vadim et al., 2018; Popescu et al., 2019; Vo et al., 2019; Yeo et al., 2021) which was built on top of Duckling². It detects temporal expressions in a given question and produces normalized values in standard YYYY-MM-DD format. For some experiment settings, it also re-writes original question by replacing original temporal expressions with their normalized values.

4.2.2 Data Variants

We create three variants (different natural language questions but the same SQL query) as follows:

Original Temporal Questions. This variant is questions having original temporal expressions

¹<https://github.com/IBM/TempQ4NLIDB-dataset>

²<https://duckling.wit.ai/>

without normalizing by DA. It is the most challenging data format for any given model to learn by mapping between original temporal expressions in a given question and the actual DB date values in the associated SQL query.

Full-DA Questions. Temporal questions that are pre-processed and re-written with Data Annotation and date values are appended at the end of questions. This variant is inspired by the mechanism that ValueNet (Brunner and Stockinger, 2021) used to learn values from given questions.

Partial-DA Questions. Temporal questions that are pre-processed and re-written with Data Annotation without date values appended at the end of questions. This variant can ease the learning of a given model by mapping between the normalized date values in a question and the DB values from its associated SQL query.

Examples of question variants and same SQL.

- *Original* question: What products were sold from 2011 to 2015?
- *Full-DA* question: What products were sold from 2011-01-01 to 2015-12-31?; 2011-01-01#date#date; 2015-12-31#date#date
- *Partial-DA* question: What products were sold from 2011-01-01 to 2015-12-31
- *SQL*: SELECT distinct T2.PRODUCT_ID FROM SALES AS T1 JOIN SALES_DETAILS AS T2 ON T1.SALES_ID=T2.SALES_ID WHERE T1.DATE >= '2011-01-01' AND T1.DATE <= '2015-12-31'

4.3 Annotation Guideline

Data annotation for Text-to-SQL task is not trivial. For every data point, we must create a pair of question and associated SQL query. In addition, data annotation for Text-to-SQL with temporal aspect is even more challenging as every question needs to have at least one new temporal dimension. We define and practise the following guideline.

4.3.1 Temporal Dimension Annotation

We define the following annotation guideline for temporal questions.

Mapping Temporal Operators to SQL. We follow the standard temporal operators {BEFORE, AFTER, ON, IN, BETWEEN...AND} in temporal questions mapping into standard SQL operators supported by SQLite for temporal aspect {<, >, >=, <=}

=, >= AND <=}. Especially, the BETWEEN operator in Spider is not inclusive and mapped into (a_date > start_date AND a_date < end_date) at SQL level. Unlike Spider, our annotation for BETWEEN at SQL level is inclusive. For example:

- Spider: What roles did staff members play between '2003-04-19' and '2016-03-15'?
SELECT role_code FROM Project_Staff WHERE date_from > '2003-04-19' AND date_to < '2016-03-15'
- Our annotation: What are employees hired between Jan 2012 and Jun 2012?
SELECT distinct EMPNAME FROM EMPLOYEE WHERE HIREDATE >= '2012-01-01' AND HIREDATE <= '2012-06-30'

In our annotation, we support two new temporal operators SINCE for after but including, and BY for before but including (Table 2).

Mapping Temporal Expressions to SQL. For mapping from temporal expressions to date values in SQL queries, we use our Date Annotator to capture and convert all temporal expressions into standard date format YYYY-MM-DD.

If we assume that time always exists as an interval with a start_point and an end_point, we can translate any temporal expression into a time range. For example:

- Christmas 2000 = [2000-12-25, 2000-12-25]
- July 2020 = [2020-07-01, 2020-07-31]
- 2021 = [2021-01-01, 2021-12-31]
- Q1 of 1999 = [1999-01-01, 1999-03-31]

Depending on the combination between temporal operator and temporal expression in a given question, we define rules to convert to SQL operators and values (Table 2).

Referring to the categories of temporal expressions in the study (Jia et al., 2018), in the scope of this paper, we only focus on using explicit temporal expressions for our dataset. We do not support implicit temporal questions (e.g. *5 years ago*, *last year*) because for annotating normalized value of implicit temporal expressions, it is required to have a time anchor which we cannot embed into our data for Text-to-SQL task. For example, given the **time anchor as 2022**, *last year* = 2021. However, when it is 2023 or later, the value 2021 is no longer correct for *last year*. Due to the nature of current data format of Text-to-SQL task, we cannot embed a specific time anchor into each question and associated SQL to support implicit temporal questions.

4.3.2 Temporal Questions and SQL Creation

Next we define the following procedure to create natural language questions and SQL queries.

Step 1: Generating simple SQL queries without temporal aspect.

We first look into a given database and generate SQL query based on the database structure for expected information. For example: in HR database, we can generate different simple queries to derive information from every column in EMPLOYEE table: SELECT EMPNO, SELECT EMPNAME, SELECT MGRNAME, SELECT BIRTHDATE, SELECT HIREDATE, SELECT LEAVEDATE, SELECT SALARY, SELECT DPNAME.

We also can generate SELECT for more than one columns, for example: SELECT EMPNO, EMPNAME, SALARY FROM EMPLOYEE. We define patterns of table names and column names to automatically generate simple SQL queries (Popescu et al., 2022).

Step 2: Identifying temporal column.

We need to verify which column has at least one temporal aspect and establish the association between them. For example:

1. There are 3 temporal columns: BIRTHDATE, HIREDATE, LEAVEDATE
2. Columns that can have association with the 3 temporal columns: EMPNO, EMPNAME
3. Without columns in (2), these columns can have no association with the 3 temporal columns: MGRNAME, SALARY, DPNAME

Now we can attach the column having temporal aspect with corresponding simple SELECT that we created in Step 1 using following pattern: [SELECT Column_from_(2) FROM EMPLOYEE WHERE Temporal_Column_in_(1) Temporal_Operator Temporal_Value]

For example: SELECT EMPNO, EMPNAME, MGRNAME FROM EMPLOYEE WHERE LEAVEDATE = '2022-05-17'

Step 3: Searching Temporal Operator and Temporal Value from DB.

For evaluation with Execution Accuracy metric, the SQL query in Step #2 needs to return a valid result from DB submission. Thus, we develop a search algorithm to find unique temporal operator and temporal value so that the SQL query will return a valid result from DB. At the end of this step, we also manually verify correctness of the result of every SQL query against the corresponding DB.

Temporal Operators and Expressions	Operators and Values in SQL
BEFORE a_temporal_expression AFTER a_temporal_expression BY a_temporal_expression SINCE a_temporal_expression ON IN a_temporal_expression BETWEEN temp_exp_A AND temp_exp_B FROM temp_exp_A TO temp_exp_B	a_date < start_point of a time range a_date > end_point of a time range a_date <= end_point of a time range a_date >= start_point of a time range a_date >= start_point AND a_date <= end_point a_date >= start_point_A AND a_date <= end_point_B a_date >= start_point_A AND a_date <= end_point_B
BEFORE July 14th 2021 BEFORE July 2021 BEFORE 2021 AFTER Oct 25th 2020 AFTER Oct 2020 AFTER 2020 SINCE Mar 20th 2018 SINCE Mar 2018 SINCE 2018 BY Apr 7th 2018 BY Apr 2018 BY 2018 ON 1/1/2021 ON Christmas 2017 IN July 2022 IN 3rd quarter of 2016 BETWEEN 1990 AND 2000 FROM 2000 TO 2010	a_date < '2021-07-14' a_date < '2021-07-01' a_date < '2021-01-01' a_date > '2020-10-25' a_date > '2020-10-31' a_date > '2020-12-31' a_date >= '2018-03-20' a_date >= '2018-03-01' a_date >= '2018-01-01' a_date <= '2018-04-07' a_date <= '2018-04-30' a_date <= '2018-12-31' a_date >= '2021-01-01' AND a_date <= '2021-01-01' a_date >= '2017-12-25' AND a_date <= '2017-12-25' a_date >= '2022-07-01' AND a_date <= '2022-07-31' a_date >= '2016-07-01' AND a_date <= '2016-07-31' a_date >= '1990-01-01' AND a_date <= '2000-12-31' a_date >= '2000-01-01' AND a_date <= '2010-12-31'

Table 2: Rules and Examples for Mapping Temporal Operators and Expressions into SQL Queries

Step 4: Generating Natural Language Questions. We manually generate natural language questions for SQL queries created in Step #3 by defining semantic relations between entities and attributes. We also enrich the language of questions by using semantic similarity/relatedness techniques (e.g. paraphrasing, synonyms) and different syntax structures (e.g. passive voice, relative clause, prepositional phrase) to generate various WH-questions (Popescu et al., 2018; Vo and Popescu, 2016; Vo et al., 2015; Vo and Popescu, 2015a,b). For examples:

- Employee {has | was born} BIRTHDATE.
- Employee {joined | is hired | is employed | is recruited | started working} in a DEPARTMENT on a HIREDATE.
- Employee {left | retired} a DEPARTMENT on a LEAVEDATE.

We carefully generate explicit temporal expressions from the temporal filters created in Step #3 and attach to our natural language questions to generate temporal questions. For examples:

1. How many employees were hired *in December 2012*?
2. Which employees left *before April 2010*?
3. What employees joined Marketing department *after Christmas 2010*?
4. Show me employees hired for Manufacturing department *in 1970* and retired *in 2000*.
5. What are the Manufacturing employees with birthdays *between 1939 and 1945*?
6. What are the employees in Sales department that have birthdays *before 1970*?

4.3.3 Data Annotation Validation

The annotation is semi-automatic and then data is manually curated by one worker. We not only examine the correctness of SQL syntax for every given question, we also submit every SQL query to corresponding DB and examine the result returned from DB. Thus, we ensure that every question always has a valid result returned from submitting its SQL query to corresponding DB. Finally the data is examined and validated by other two workers.

5 Experiments with SOTA Models

We define three settings to experiment with two recent SOTA models: ValueNet (Brunner and Stockinger, 2021) and Picard (Scholak et al., 2021) for temporal questions.

Setting 1: Original Models trained on Spider.

We evaluate original ValueNet and Picard models (that were trained only on Spider dataset) on temporal questions in Test set of our new dataset.

Setting 2: Only Temporal Questions. We train and evaluate ValueNet and Picard (which were already pretrained on Spider) on temporal questions in Train and Test set of our dataset, respectively.

Setting 3: Blended Questions (NT+Full-DA). We mix Full-DA temporal questions with other Non-Temporal questions of the same database for training (on top of Spider as in Setting 2) and test.

Picard. We fine-tuned T5-large with the data splits described in settings 1 through 3 above for each DB in the TempQ4NLIDB dataset. For settings 2 and 3, we fine-tuned first on the Spider dataset, took the best performing model, and continue the training on the corresponding training data combination using the temporal data for each schema. Each model was fine-tuned on 448 epochs. Training the model for more epochs did not improve the model performance on the validation set. We used Adafactor (Shazeer and Stern, 2018), a learning rate of 10^{-5} , and a batch size per device of 5. During testing, we enabled Picard with the highest parsing mode.

Table 3 shows evaluation results of HR and WH temporal test sets with Picard. For Setting 1, models trained without temporal data (using the default Spider dataset for training) show very poor understanding of temporal questions. For Setting 2, models trained on temporal data understand temporal questions much better. Setting 3 results show a declining performance for models to handle both temporal and non-temporal questions.

ValueNet (VL). We use the default configuration for VL experiments. VL evaluation only reports execution accuracy. For Setting 1 and 2, the VL models performed poorly on the temporal questions. The model trained on the Spider dataset only, but also adding the available trying, obtained less than 1% accuracy. The main reasons is that the VL encoder receives no information about the type of the columns and values, and when there are more than two values in the SQL query, the system systematically confound them. Because VL does not

implement any control over the correctness of SQL formula, many of the SQL queries with multiple values are wrong because the values are switched between themselves or the wrong operator is used, like "=" instead of ">=". As the temporal questions are at least two values with multiple operators the inferred SQL was always wrong. For Setting 3 experiments (Table 4), as some of the questions themselves were not multi-values, the accuracy was significantly higher.

Observations. We learn the following lessons:

1. Only training on Spider is insufficient for understanding temporal condition questions.
2. Additional training on TempQ4NLIDB significantly helps models to improve the understanding of temporal condition questions.
3. It is challenging for models to understand temporal expressions in given questions then generate corresponding temporal filters with normalized values (error type 2). Rewriting questions with normalized values of temporal expressions (Partial-DA and Full-DA variants) will help to generate temporal filters with correct values.
4. We mix temporal and non-temporal questions for both training and testing to increase the complexity (e.g. multi-table joins, multiple selects, multiple values/filters, more complex language and sentence structure, etc). It is more challenging to handle both temporal and non-temporal questions than just one type.
5. More works are needed to expand the coverage for other types of temporal questions (e.g. implicit one).

6 Error Analysis

We present four error types in predictions made by Picard and ValueNet for temporal questions.

Type 1. Models cannot detect temporal values in question, thus, no corresponding filter created.

- How many iphones were sold *since October 2013* in shops located in New York?

Prediction: `select sum(t2.quantity) from products as t1 join sales_details as t2 on t1.product_id = t2.product_id join sales as t3 on t2.sales_id = t3.sales_id join shops as t4 on t3.shop_id = t4.shop_id where t4.address = 'New York' and`

Setting 1			
DB	Test	Match(%)	Exec(%)
HR	Original	4.35	0.00
	PartialDA	0.00	0.00
	FullDA	0.00	0.00
WH	Original	3.57	10.71
	PartialDA	0.00	10.71
	FullDA	0.00	10.71
Setting 2			
HR	Original	82.61	82.61
	Partial-DA	95.65	95.65
	Full-DA	100.00	100.00
WH	Original	67.86	60.71
	Partial-DA	89.29	92.86
	Full-DA	89.29	92.86
Setting 3			
HR	NT + Full-DA	58.42	66.34
WH	NT + Full-DA	72.00	75.00

Table 3: Picard’s Performance for Setting 1, 2, and 3.

Setting 3 (ValueNet)			
DB	Test	Match(%)	Exec(%)
HR	NT + Full-DA	N/A	47%
WH	NT + Full-DA	N/A	39%

Table 4: ValueNet’s Performance for Setting 3.

t1.type = 'IPHONE'

Actual: select distinct sum(T3.QUANTITY) from SHOPS AS T1 JOIN SALES AS T2 on T1.SHOP_ID = T2.SHOP_ID JOIN SALES_DETAILS AS T3 on T2.SALES_ID = T3.SALES_ID JOIN PRODUCTS AS T4 on T3.PRODUCT_ID = T4.PRODUCT_ID where T2.DATE >= '2013-10-01' and T4.TYPE = 'IPHONE' and T1.ADDRESS = 'New York'

Type 2. Original temporal expressions used in predicted query without normalizing values.

- Who were recruited in Marketing department since March 2012?

Prediction: select distinct empname from employee where dptname = 'Marketing' and hiredate > 'March 2012'

Actual: select distinct EMPLOYEE.EMPNO from EMPLOYEE where EMPLOYEE.DPTNAME = 'Marketing' and EMPLOYEE.HIREDATE >= '2012-03-01'

Type 3. Missing temporal filters in prediction in case question has multiple temporal expressions.

- What are the names of Sales employees who were hired after 1977-12-31 and departed before 2020-01-01; 1977-12-31#date#date; 2020-01-01#date#date

Prediction: SELECT DISTINCT T1.empname FROM employee AS T1 WHERE T1.dptname = '1977-12-31' and T1.hiredate < 'Sales'

Actual: select distinct EMPLOYEE.EMPNAME from EMPLOYEE where EMPLOYEE.HIREDATE > '1977-12-31' and EMPLOYEE.LEAVEDATE < '2020-01-01' and EMPLOYEE.DPTNAME = 'Sales'

Type 4. Incorrect filters (matching with wrong DB columns) created for temporal values.

- How many iphones were sold since 2013-10-01 in shops located in New York; 2013-10-01#date#date

Prediction: SELECT DISTINCT sum(DISTINCT T1.quantity) FROM sales_details AS T1 JOIN products AS T2 ON T1.product_id = T2.product_id JOIN stock AS T15 ON T2.product_id = T15.product_id JOIN shops AS T3 ON T15.shop_id = T3.shop_id JOIN sales AS T4 ON T3.shop_id = T4.shop_id WHERE T2.type = '2013-10-01' and T3.address = '2013-10-01' and T4.date = '2013-10-01'

Actual: select distinct sum(T3.QUANTITY) from SHOPS AS T1 JOIN SALES AS T2 on T1.SHOP_ID = T2.SHOP_ID JOIN SALES_DETAILS AS T3 on T2.SALES_ID = T3.SALES_ID JOIN PRODUCTS AS T4 on T3.PRODUCT_ID = T4.PRODUCT_ID where T2.DATE >= '2013-10-01' and T4.TYPE = 'IPHONE' and T1.ADDRESS = 'New York'

7 Conclusions and Future Work

In this paper, we attempt to tackle the temporal aspect in NLIDB. This is a practical and challenging topic in real-world applications for which there are not many studies. Being motivated by the necessity and lack of available dataset for temporal questions, we created the new dataset TempQ4NLIDB. We also experimented with two SOTA models in NLIDB and show that they benefit from our dataset for better learning temporal questions. In future, we will increase the size of our dataset and expand different types of temporal questions (e.g. implicit).

References

- Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases—an introduction. *Natural language engineering*, 1(1):29–81.
- Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Representing schema structure with graph neural networks for text-to-sql parsing. *arXiv preprint arXiv:1905.06241*.
- Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 2177–2182. IEEE.
- Ruichu Cai, Boyan Xu, Xiaoyan Yang, Zhenjie Zhang, Zijian Li, and Zhihao Liang. 2018. [An encoder-decoder framework translating natural language to database queries](#).
- Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.
- Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning.
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.
- Christian S Jensen and Richard T Snodgrass. 2018. Temporal database.
- Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Janik Strötgen, and Gerhard Weikum. 2018. Tempquestions: A benchmark for temporal question answering. In *Companion Proceedings of the The Web Conference 2018*, pages 1057–1062.
- Yaghmazadeh Navid, Wang Yuepeng, Dillig Isil, and Thomas Dillig. 2017. Query synthesis from natural language. *International Conference on Object-Oriented Programming*.
- Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases.
- Octavian Popescu, Ngoc Phuoc An Vo, Vadim Sheinin, Elahe Khorashani, and Hangu Yeo. 2019. Tackling complex queries to relational databases. In *Asian Conference on Intelligent Information and Database Systems*, pages 688–701. Springer.
- Octavian Popescu, Irene Manotas, Ngoc Phuoc An Vo, Hangu Yeo, Elahe Khorashani, and Vadim Sheinin. 2022. [Addressing limitations of encoder-decoder based approach to text-to-SQL](#). In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1593–1603, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.
- Octavian Popescu, Ngoc Phuoc An Vo, and Vadim Sheinin. 2018. [A large resource of patterns for verbal paraphrases](#). In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).
- Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of the 55th Annual Meeting of the 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.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. [PICARD: Parsing incrementally for constrained auto-regressive decoding from language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.
- Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.
- Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2019. Semantic parsing with syntax and table aware sql generation. *56th Annual Meeting of the Association for Computational Linguistics*.
- Sheinin Vadim, Khorasani Elahe, Yeo Hangu, Xu Kun, An Vo Ngoc, Phuoc, and Octavian Popescu. 2018. Quest: A natural language interface to relational databases. *LREC*.
- Ngoc Phuoc An Vo, Simone Magnolini, and Octavian Popescu. 2015. [FBK-HLT: A new framework for semantic textual similarity](#). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 102–106, Denver, Colorado. Association for Computational Linguistics.
- Ngoc Phuoc An Vo and Octavian Popescu. 2015a. [Learning the impact and behavior of syntactic structure: A case study in semantic textual similarity](#). In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 688–696, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.
- Ngoc Phuoc An Vo and Octavian Popescu. 2015b. [A preliminary evaluation of the impact of syntactic structure in semantic textual similarity and semantic relatedness tasks](#). In *Proceedings of the 2015*

Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 64–70, Denver, Colorado. Association for Computational Linguistics.

Ngoc Phuoc An Vo and Octavian Popescu. 2016. A multi-layer system for semantic textual similarity. In *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 56–67.

Ngoc Phuoc An Vo, Octavian Popescu, Vadim Sheinin, Elahe Khorasani, and Hangu Yeo. 2019. A natural language interface supporting complex logic questions for relational databases. In *International Conference on Applications of Natural Language to Information Systems*, pages 384–392. Springer.

Ziyu Yao, Yu Su, Huan Sun, and Wen tau Yih. 2010. Model-based interactive semantic parsing: A unified framework and a text-to-sql case study. *IJCNLP*.

Hangu Yeo, Elahe Khorasani, Vadim Sheinin, Ngoc Phuoc An Vo, Octavian Popescu, and Petros Zefos. 2021. Programmatic database language generation for big data applications. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 2848–2856. IEEE.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. [Neural enquirer: Learning to query tables with natural language](#).

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based sql query generation for cross-domain context-dependent questions. *arXiv preprint arXiv:1909.00786*.

Multi-Tenant Optimization For Few-Shot Task-Oriented FAQ Retrieval

Asha Vishwanathan
Verloop.io
asha@verloop.io

Rajeev Unnikrishnan Warriar
Verloop.io
rajeev@verloop.io

Gautham Vadakkekara Suresh*
Verloop.io
gautham.suresh09@yahoo.com

Chandra Shekhar Kandpal*
Verloop.io
chandrashekhar1503@gmail.com

Abstract

Business-specific Frequently Asked Questions (FAQ) retrieval in task-oriented dialog systems poses unique challenges vis-à-vis community based FAQs. Each FAQ question represents an intent which is usually an umbrella term for many related user queries. We evaluate performance for such Business FAQs both with standard FAQ retrieval techniques using query-Question (q-Q) similarity and few-shot intent detection techniques. Implementing a real-world solution for FAQ retrieval in order to support multiple tenants (FAQ sets) entails optimizing speed, accuracy and cost. We propose a novel approach to scale multi-tenant FAQ applications in real-world context by contrastive fine-tuning of the last layer in sentence Bi-Encoders along with tenant-specific weight switching¹.

1 Introduction

Business-specific Frequently Asked Questions (FAQ) form an important part of many task-oriented dialog systems today. Business FAQs exhibit some common characteristics which differentiate them from community-based FAQs. We show few examples of such FAQs in Table 1. The intent is often a user-defined umbrella term that represents a group of questions in such FAQs. The system needs to respond to user queries which indicate the same intent as that of the FAQ question (Example-1). In some cases, similar FAQ questions can lead to different answers (Example-2). The answers may not always have overlap with the question (Example-3). The response can also be modeled as a system action rather than a text response (Example-4). The system should also be able to identify Out-Of-Scope (OOS) queries and redirect to a human agent.

Since each question within an intent category represents it loosely, having a single question for

an intent poses a challenge for intent detection. Simple paraphrases do not suffice and therefore, 5-10 variations for each FAQ question representing the intent is sourced from domain experts.

Common approaches in FAQ retrieval include Query-Question similarity and Query-Answer entailment (Sakata et al., 2019; Hammond et al., 1995; Tomuro and Lytinen, 2004). Query-Question (q-Q) methods focus on similarity between a user’s query and a question to retrieve the relevant answer. Query-Answer (q-A) methods predict the relevance between the query and an FAQ answer to re-rank the results obtained via q-Q. For FAQ retrieval, we leverage q-Q similarity. The q-A entailment method is infeasible in some FAQ sets due to the generic nature of answers for several different questions, owing to little match between the q-A pair and responses being modeled as system action instead of text.

We also prefer a single model based approach over stacked models such as retriever-reranker models (Zhang et al., 2020; Gupta and Carvalho, 2019) as inference latency gets compounded in such systems. We compare several q-Q retrieval approaches using pre-trained models and fine-tuned embeddings. We also model the problem as a few-shot intent detection and evaluate classifier based techniques.

From an industry specific dialog solutions perspective, the approaches need to balance accuracy, real time inference latency and costs while having the ability to scale for multiple tenant (FAQ sets) requirements. Our contribution is towards implementing an FAQ retrieval system for Business FAQs in real-world contexts. We describe a low-cost approach to deploy multi-tenant FAQ applications at scale while optimizing for inference latency and accuracy. We also evaluate different methods and models against few-shot intent detection and conversational datasets.

*Work done while the authors were working at Verloop.io

¹<https://github.com/verloop/few-shot-faqir>

No.	Question	Answer
1	How to check exchange rate? Rate for Peso	Please use the Rate Enquiry option on the Main Menu
2	Am I going to get a refund When am I going to get the refund	We issue refund for specific products Refund normally takes 7 days to process
3	My transaction failed My checkout did not happen properly	Please contact on XXX or mail us at XXX
4	Can I speak to an agent?	System action: Transfer to agent

Table 1: Examples of actual business questions. All identifying information is masked.

1.1 Real world constraints

We process millions of chat messages every day, of which a sizable percentage is related to FAQ retrieval. The expected latency requirement is less than 100 ms under production loads. Tenant specific FAQ sets vary in terms of the domain and intent granularity. It is important to maximize the retrieval accuracy for each tenant while not increasing inference costs drastically.

As on-premise machines are expensive to set-up and maintain, it is often cost-effective and convenient to leverage cloud providers. The application efficiently needs to utilize machine resources and scale for multiple tenants without significant overhead. Since GPU costs are higher, it is cost-effective to train using GPU machines and perform inference with CPU machines.

Apart from this, many tenants require the system to respond with certain standardized answers based on company policy without too many modifications. Therefore, retrieval based solutions are more acceptable than generative solutions. The problems of hallucinations which still exist in generative solutions (Roller et al., 2021; Shuster et al., 2021) constraints the use of these approaches for task-oriented FAQ responses in business contexts.

2 Related Work

The models based on transformer architectures have enabled the usage of transfer learning for various tasks such as text similarity (Devlin et al., 2019). Sakata et al. (2019) introduced an approach to combine the query-question similarity from TSUBAKI and query-answer relevance from BERT. Arora et al. (2020) evaluated different providers on three intent recognition datasets, created using queries received by chatbots from real users.

In a few-shot setting, recent approaches leverage the transformer architecture. Casanueva et al. (2020) demonstrated that using a dual sentence en-

coder as base model works better in low-resource intent detection tasks as compared to BERT. Zhang et al. (2021b) proposed the use of contrastive learning to improve the performance of BERT based classifiers in few-shot intent detection tasks.

Pre-training methods for creating transferable language representations is a common approach. In-domain pre-training further increases the adaptability of the model to domain related downstream tasks. Specifically for few-shot setting, fine-tuning pre-trained models with supervised learning is found to be effective (Zhang et al., 2021a).

Cross-encoder approaches with BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models have achieved SOTA scores using sentence pair training. Training is done with question pairs. During inference, question pairs are created using query and train questions. These pairs are scored by a classifier to predict the final label.

Bi-Encoder models have the advantage of being able to cache the representations and hence, are highly efficient during inference (Reimers and Gurevych, 2019). The research on STILTS (Phang et al., 2018) found the advantages of intermediate training on data rich supervised tasks, giving pronounced benefits in data constrained regimes. Casanueva et al. (2020) discusses the gains achieved via transfer learning using pre-trained sentence encoders as a retriever model. Cross-Encoder models entail a higher latency as all the question pairs are required to be classified during inference. Zhang et al. (2020) explores usage of the classifier model as a re-ranking model following a Bi-Encoder based retriever to reduce inference time.

We evaluate our FAQ retrieval use case with simple baseline approaches, in-domain pre-trained models, classifier based approaches, Cross-Encoders and Bi-Encoders in a few-shot setting.

3 Datasets

We choose datasets that are similar to our real-world tenant FAQs and perform experiments on publicly available intent detection datasets for task-oriented dialogues. We use datasets from HINT3 (Curekart, Powerplay11 and SOFMattress) (Arora et al., 2020) which reflects real-world FAQ intents and user queries. We also use DialoGLUE datasets (Mehri et al., 2020) - HWU64 (Liu et al., 2021), CLINC150 (Larson et al., 2019) and BANKING77 (Casanueva et al., 2020).

DialoGLUE datasets further provide train sets for data constrained regimes with 5 and 10 samples per class. Similarly, HINT3 also contains subset variants of datasets which are created by retaining only samples whose entailment score using the ELMo model was less than 0.6. We use the few-shot subsets for DialoGLUE and the full as well as subset variant of HINT3 datasets. These datasets are suitable for few-shot analysis. All the datasets have separate train and test sets and we report our evaluations on the test sets. The CLINC150 and HINT3 datasets have separate in-scope and out-of-scope queries and we use them to evaluate OOS performance. The dataset size, intent and domain distribution is listed in Appendix A.

4 Methodology

In this section, we elaborate on our experiments for model training and deployment. We compare different approaches of few-shot FAQ retrieval and describe our deployment strategy for the same.

4.1 FAQ Retrieval approaches

4.1.1 Baseline

We consider BM25 (Mass et al., 2020; Karan and Šnajder, 2016) and vector space using TF-IDF (Karan and Šnajder, 2016) as baseline approaches.

4.1.2 Pre-trained and fine-tuned features

The following models are considered for feature extraction:

- Pre-trained BERT
- Pre-trained ConvBERT: ConvBERT model is pre-trained on open-domain conversational data (Mehri et al., 2020).
- Fine-tuned ConvBERT: We further fine-tune ConvBERT model as an intent classifier and use the fine-tuned encoder as a feature extractor.

- Pre-trained Bi-Encoders: We use pre-trained Sentence BERT models (*mini-LM-L6-v2*, *all-mpnet-base-v2*) which are trained on more than 1 Billion sentence pairs consisting of a diverse set of duplicate question pairs, NLI sets, QA pairs.
- Fine-tuned Bi-Encoders: We fine-tune pre-trained Bi-Encoders for a Semantic textual similarity task with a contrastive loss (Zhang et al., 2021b).
- Fine-tuned Task adapted pre-trained Bi-Encoders: We perform pre-training for adaptation of pre-trained Bi-Encoders with a triplet loss using a cosine distance metric and further fine-tune using a contrastive loss

After the features are extracted, inference is implemented using a cosine similarity of query and question embeddings.

4.1.3 Classifier

We fine tune BERT with a linear layer on top to predict the intent class directly (Mehri et al., 2020).

4.1.4 Cross-Encoders

Inspired by the success of Cross-Encoders and the STILTS approach, we fine-tune a pre-trained Cross-Encoder model from SBERT (*sts-distilroberta-base*).

4.2 Question Pair/Triplet Based fine-tuning

We form Question pairs to fine-tune all Bi-Encoder and Cross-Encoder models. We mark the question pairs belonging to same class as positive samples and ones belonging to different classes as negative samples.

If $c \in C$ Where C is the set of all classes, then we denote $q_{c,i}$ to be the i^{th} element of the c^{th} class. Then, $q_{c,i} \in Q$ where Q is the set of all Questions. $i \in \{1, \dots, n\}$ where $n \in \{5, 10, N\}$ when testing for few-shot training with 5 samples, 10 samples or the entire set respectively. The question pairs are labelled as defined below.

$$L(q_{l,j}, q_{m,k}) = \begin{cases} 1 & \text{if } l = m \\ 0 & \text{if } l \neq m \end{cases} \quad (1)$$

$$\forall l \neq m \text{ and } j \neq k$$

This method increases the data size as it generates ${}^n C_2$ question pairs given n questions. Even in a few-shot setting where there are only 180 samples across 21 intents (SOFMattress), 16110 question

pairs were generated. The Q-Q data can grow exponentially, so the train data is capped to 200K for fine-tuning. We also check fine-tuning on 50K Balanced samples. Hard sampling is done using a probabilistic method based on sample weights. For Q-Q pair with label 0, the weight is equal to the cosine similarity between the Q-Q pair. For label 1, the weight is $(1 - \text{cosine similarity})$ of the Q-Q pair. We use the same pre-trained bi-encoder model for the embeddings which we intend to fine-tune. Sampling with replacement is then done for labels 0 and 1 to get the required sample sizes. For the triplet based pre-training, the triplets are constructed such that the anchor and positive belong to the same label and the negative belong to a different label. Q-Q pairs are first constructed using the hard sampling approach described above. For each sample which is the triplet anchor, we get its positives and negatives from the Q-Q pairs. Sampling is then done based on weightage to construct the triplets.

4.3 Training and deployment

The common approach for fine-tuning models involves modifying weights in all layers of the pre-trained model. Instead, we propose an approach where only the final layer of the model is fine-tuned. The weights of all layers which are not fine-tuned are shared across tenant models. During inference in the production environment, we load the base model only once for shared parameters. We keep the tenant-specific weights for all tenants loaded in memory and we switch the tenant specific weights in the model for every inference request. The shared base model weights reduce the memory requirement by a significant margin when we scale to a large number of tenants. Since we retain the tenant weights in memory, replacing the model weights does not result in any significant latency overhead. This allows us to support multiple tenants using the same machine and in-turn reduces the inference costs by an order based on the model size. We select the best model based on expected number of tenants, throughput and memory requirement. We evaluate the performance of all fine-tuned approaches under the following constraints: Freezing all layers of the encoder except the last layer, training for a fixed number of iterations and using few-shot samples.

5 Experimentation Approaches

Fine-tuning for all Bi-Encoder and Cross-Encoder models is done using question pairs as elaborated in Section 4. Based on the training method outlined by Mosbach et al. (2021), we train for a higher number of iterations to offset any fine-tuning instability. For the Bi-Encoders, we fine-tune only the final layer of the model using a contrastive or an online triplet loss. The pre-trained BERT and ConvBERT classifier models were fine-tuned using a softmax cross-entropy loss. The Cross-Encoder was also fine-tuned with the same approach using question pairs with a binary cross-entropy loss. For predicting the query label, we create q-Q pairs limited to 5 questions per intent. The model predicts a score for these combinations, where the label for the question with the highest score is considered to be the predicted label. We trained all models for 10 K iterations with a learning rate of $2e-5$, batch size of 16, AdamW optimizer with 10% linear warm up and gradient normalization.

5.1 Pre-training

Commonly pre-training is done for in-domain adaptation using unlabelled datasets. But in our case, tenant FAQs are spread across multiple niche domains, making it difficult to get domain related data. Moreover, pre-training for each domain separately would result in multiple models per domain, making the hosting costs higher. In case of pre-trained dense retrievers, training on the base language model in an MLM fashion requires a further training of the bi-encoder model. In the GPL paper, Wang et al. (2021) adopt a pre-training method which is a triplet based training where the triplets are constructed from the unlabelled data. Gururangan et al. (2020) describe an approach of a second level pre-training using unlabelled corpus and pre-training using task related samples within a domain.

We experiment with a similar approach as GPL but we use the labelled data for the same task available across different domains. We construct triplets using an offline approach instead of an online batch mode, to ensure in-domain triplets. We create 100 K triplets from each dataset and get a total training corpus of 600 K samples. We do a second level of pre-training of the pre-trained bi-encoder models for 140 K iterations, with the Triplet loss using Cosine distance metric and a margin of 0.15. We then use this pre-trained model for further finetuning

Method	BANKING77		CLINC150		HWU64	
	5	10	5	10	5	10
BM25	53.96	61.10	55.37	60.80	50.37	54.46
TF-IDF	49.51	58.14	60.91	58.14	60.91	54.36
BERT - MP	40.25	48.89	68.97	66.73	52.88	57.06
ConvBERT - MP	50.42	59.41	73.11	79.53	58.92	65.79
ConvBERT - FT(C)	56.98	66.91	76.44	82.91	64.86	71.46
SBERT (MiniLM-L6)	76.78	83.47	79.08	81.24	68.02	73.42
SBERT (MiniLM-L6) - FT(C)	80.74	86.00	84.55	87.13	76.20	79.73
SBERT (MiniLM-L6) - FT(C) 50K	76.42	83.18	83.97	86.86	73.97	77.60
SBERT (MiniLM-L6) - FT(T)	81.33	86.33	85.11	87.75	75.27	82.24
SBERT (MiniLM-L6) - PT-FT(C)	84.28	84.67	89.88	89.86	85.68	86.24
SBERT (MPNet) - FT(T)	83.21	88.18	88.68	91.00	78.06	83.05
SBERT (MPNet) - PT-FT(C)	86.98	87.27	92.51	92.68	86.24	85.5
BERT - FT(C)	22.13	23.64	17.35	15.12	39.98	41.13
SBERT Cross-Encoder - FT(C)	67.10	69.83	76.10	75.20	66.91	68.03

Table 2: Top-1 accuracy of models on the DialoGLUE test sets. MP stands for Mean-Pooling, FT for fine-tuning, (C) for Contrastive loss, (T) for Triplet loss and PT for pre-training. Here, 5 and 10 refers to training subsets created with 5 and 10 samples per intent, respectively.

with a contrastive loss.

6 Evaluation Metrics

For evaluation, we consider a combination of FAQ Retrieval metrics along with Intent detection metrics. Similar to the metrics used in Sakata et al. (2019), we consider Top-k accuracy (same as Success Rate referred in the paper), MRR@k (Mean Reciprocal Rank), nDCG@k (normalized Discounted Cumulative Gain) and MAP@k (Mean Average Precision). From an intent detection perspective, we also evaluate out-of-scope accuracy across different thresholds. We use accuracy as the metric for evaluation of OOS queries. In case the top predicted label score is less than a specified threshold, then the top-k results are shown as suggestions to the user. Metrics like MRR, nDCG and MAP put emphasis on the ordering of the final responses and hence are useful for evaluating the effectiveness of this approach.

7 Results

Table 2 and 3 show the Top-1 accuracy for all experiments. Table 4 shows the performance of the MiniLM-L6 model in top-3 setting. Figure 1 further shows its OOS and in-scope accuracy against different thresholds.

From our experiments, we find that the fine-tuned Bi-Encoder models perform the best across all datasets. All the Bi-Encoder embedding ap-

proaches significantly outperform the other approaches in few-shot setting in spite of being constrained with the last layer fine-tuning strategy.

For HINT3 datasets, we use the benchmarking results as reported by Arora et al. (2020) for comparison. We find that the fine-tuned Bi-Encoder shows better performance than all benchmarked chatbot solutions on both full and constrained datasets. We noticed improvement while fine-tuning the pre-trained Bi-Encoder models using the Question pair approach. We also note that online triplet loss based fine-tuning performed marginally better as compared to contrastive Loss for all the base models other than MPNet model on Haptik dataset. We observe that the question pair / triplet training strategy mitigates the effect of fewer samples. Reducing sample sizes of the Q-Q pairs from 200 K to 50 K balanced samples hurts the performance across all datasets.

Fine-tuning the task adapted pre-trained models shows much better results in comparison to the base pre-trained models in most of the Dialogue datasets but the gains are either less or deteriorate for Haptik datasets. In Haptik datasets, the median examples per intent is quite low as compared to Dialogue, with some intents having only 1 sample. There would be no triplets formed for such samples. We also see a wide variation in the number of examples per intent ranging from 3 to 95 in the Curekart dataset. This would reduce the number of

Method	Curekart		Powerplay11		SOFMattress	
	Full Set	Subset	Full Set	Subset	Full Set	Subset
BM25	72.34	71.20	51.63	49.09	58.44	52.24
TF-IDF	72.56	70.35	53.81	52.72	59.74	54.54
BERT - MP	52.87	50.21	30.54	26.18	38.09	32.03
ConvBERT - MP	65.92	63.27	42.18	37.45	49.35	47.18
ConvBERT - FT(C)	77.40	75.22	46.9	43.6	62.33	56.27
SBERT (MiniLM-L6)	82.52	79.64	64.00	62.54	74.58	71.42
SBERT (MiniLM-L6) - FT(C)	87.38	86.06	64.00	63.2	78.78	77.48
SBERT (MiniLM-L6) - FT(C) 50K	86.28	86.94	61.45	58.9	75.32	74.89
SBERT (MiniLM-L6) - FT(T)	86.50	85.17	61.80	64.36	77.90	74.02
SBERT (MiniLM-L6) - PT-FT(C)	85.39	85.39	62.54	61.81	73.16	73.59
SBERT (MPNet) - FT(T)	85.61	84.51	66.54	64.00	74.45	73.59
SBERT (MPNet) - PT-FT(C)	88.05	87.83	62.18	61.81	75.32	76.19
BERT - FT(C)	55.75	58.63	18.18	18.91	41.56	38.96
SBERT Cross-Encoder - FT(C)	65.48	67.92	51.27	49.09	58.40	60.60
Dialogflow	75.00	71.20	59.60	55.60	73.10	65.30
Rasa	84.00	80.50	49.00	38.50	69.20	56.20
LUIS	59.30	49.30	48.00	44.00	59.30	49.30
Haptik	72.20	64.00	66.50	59.20	72.20	64.00
BERT Full Training	73.50	57.50	58.50	53.00	73.50	57.10

Table 3: Top-1 accuracy of models on HINT3(v1) test sets with a threshold of 0.1. Here, MP stands for Mean-Pooling, FT for fine-tuning, (C) for Contrastive loss, (T) for Triplet loss and PT for pre-training.

triplets for such intents.

While Cross-Encoder strategies are supposed to be superior, we find that the Cross-Encoder fine-tuning suffers due to the restrictive training strategy and performs sub-par when compared to the Bi-Encoders.

The BERT based classifier also shows poor performance which appears to degrade on datasets with more classes. In comparison to training BERT model with all layers unfreezed as reported in the [Arora et al. \(2020\)](#), we see that the last layer fine-tuning strategy severely impacts the model performance.

We also observe that models pre-trained on in-domain data such as ConvBERT is superior to base BERT even as a feature extractor. Interestingly, we see that even baseline approaches with BM25 and TF-IDF show better results than pre-trained BERT in such settings. Fine-tuning the ConvBERT model on the supervised classification task shows an improvement over the pre-trained model. If we look at the OOS accuracy for the fine-tuned MiniLM-L6 model, the similarity scores are normally high and lower thresholds do not have much impact. We further observe that each dataset has different thresholds where the best trade-off between OOS

and in-scope accuracy is achieved. The top-3 accuracy of MiniLM-L6 on all datasets including the challenging Powerplay11 dataset is above 70%. We choose the fine-tuned Bi-Encoder models based on superior performance with our training strategy.

8 Deployment

Smaller models give us more scalability at the cost of accuracy while larger models tend to have lower throughput and higher memory requirements. The model selection is done based on the specific business need which dictate these three metrics. Table 5 shows the load testing results using Locust framework, where we measure the mean latency per request and the request per second (RPS) achieved. The models were tested under increasing loads until RPS was stagnant and latency shot up. The results depict the performance at the maximum load the model can comfortably handle. The models are hosted via Kubernetes and deployed as a microservice using native Pytorch inference. Machine configuration used for locust testing was c2-standard-4 machine (4 vCPU and 16GB RAM) on Google Cloud Platform. From Table 6, we find that weight switching saves 81.5% memory per tenant for MiniLM-L6. For the larger model MPNet,

Metrics	Curekart		Powerplay11		SOFMattress		CLINC150	
	Full	Subset	Full	Subset	Full	Subset	Sample-5	Sample-10
Success Rate	89.80	88.93	73.81	73.09	81.38	81.81	94.22	95.08
MRR	88.38	87.38	68.42	67.45	79.94	79.43	88.88	90.77
nDCG	89.33	88.69	72.74	71.48	81.06	81.33	93.09	94.33
MAP	88	87	68	67	80	79	89	91

Table 4: Top-3 accuracy of fine-tuned MiniLM-L6 on the OOS datasets

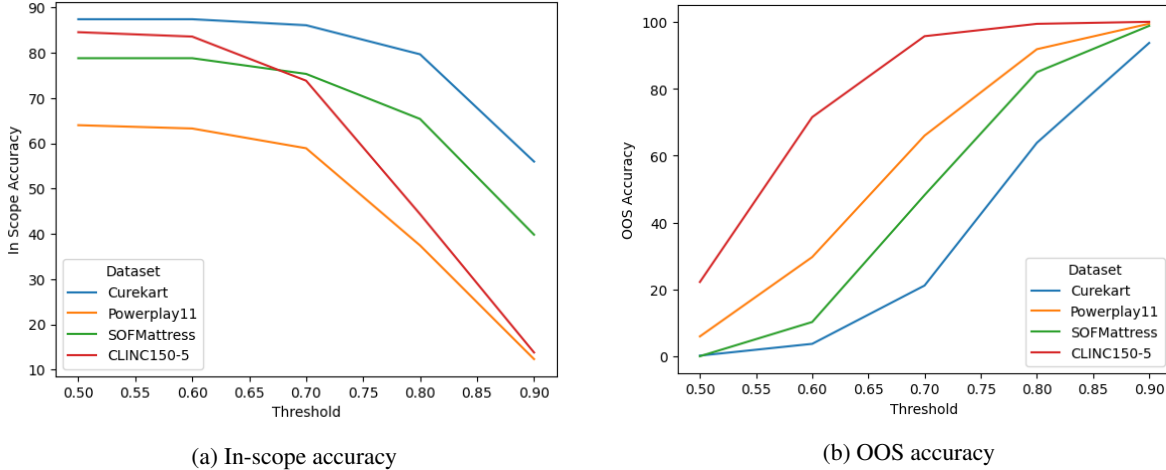


Figure 1: In-scope and OOS accuracy on different datasets for fine-tuned MiniLM-L6 model.

it saves 92.5% per tenant. The benefits of weight switching increases as model sizes gets larger. We see that fine-tuned Bi-Encoder models work best in terms of accuracy in few-shot setting. We also have a latency benefit as the question embeddings are computed upfront and cached. At inference, prediction are done as cosine similarity between the query embeddings and the cached question embeddings. For further gains, we cache these tenant specific embeddings using FAISS or ANNOY². While the task adapted pre-trained models worked quite well in terms of accuracy, it entails repeated pre-training of the model for every new tenant from a niche domain and the subsequent fine-tuning. It poses a hindrance in decoupling tenant data seamlessly. However, it is still a very viable approach in case of separate hosting of domain specific models.

Model	Median latency	90%ile latency	RPS
MiniLM-L6	93	260	22.9
MPNet	210	500	6.2

Table 5: Load testing results with latency and requests per second (RPS) for best models. All latency values in milliseconds (ms).

Model	Memory requirement	
	Full model	Weight switch
MiniLM-L6	110	20.5
MPNet	560	41.5

Table 6: Memory increase per tenant for the best performing models. All values are in MegaBytes (MBs)

We choose MiniLM-L6 as it handles higher load at significantly lower latency as compared to the best performing model MPNet, with a slight trade-off in accuracy.

9 Conclusion

We evaluated various methods for retrieval of Business FAQs by modeling the problem as an FAQ retrieval and a few-shot intent detection task. We proposed a realistic multi-tenant deployment solution with trade-offs in accuracy while balancing for cost and latency. Our last layer weight-switching strategy works well where the model has to be fine-tuned on tenant specific tasks. We observed that fine-tuned Bi-Encoder embeddings work best in a few-shot setting even under a constrained fine-tuning strategy.

²<https://github.com/spotify/annoy>

10 Limitations

Although the weight switching approach is highly scalable with increase in number of tenants, it has few limitations. Our last-layer weight switching strategy is more effective for heavier models and the benefit starts to diminish with lighter models. Although this approach is highly scalable, implementing it for inference frameworks such as ONNX Runtime and NVIDIA Triton is not straightforward. Hence, we are currently limited to using native PyTorch inference which has higher latency. Apart from this, fine-tuning only the last layer also constraints the model training leading to lower accuracy as compared to training all the layers. All our experiments showcase benefits only with base models and do not showcase inference benefits using other strategies such as quantization, pruning, ANNs and caching. Such strategies can make the system more scalable.

11 Acknowledgements

This work was supported by Verloop.io. We wish to thank Gaurav Singh and Peeyush Jain from Verloop. We thank the anonymous reviewers whose thoughtful comments improved our final work.

References

- Gaurav Arora, Chirag Jain, Manas Chaturvedi, and Krupal Modi. 2020. [HINT3: Raising the bar for intent detection in the wild](#). In *Proceedings of the First Workshop on Insights from Negative Results in NLP*, pages 100–105, Online. Association for Computational Linguistics.
- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. [Efficient intent detection with dual sentence encoders](#). In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45, 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.
- Sparsh Gupta and Vitor R. Carvalho. 2019. [Faq retrieval using attentive matching](#). In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'19*, page 929–932, New York, NY, USA. Association for Computing Machinery.
- 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.
- K. Hammond, R. Burke, C. Martin, and S. Lytinen. 1995. [Faq finder: a case-based approach to knowledge navigation](#). In *Proceedings the 11th Conference on Artificial Intelligence for Applications*, pages 80–86.
- Mladen Karan and Jan Šnajder. 2016. [Faqir – a frequently asked questions retrieval test collection](#). In *Text, Speech, and Dialogue*, pages 74–81, Cham. Springer International Publishing.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. [An evaluation dataset for intent classification and out-of-scope prediction](#). 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 1311–1316, Hong Kong, China. Association for Computational Linguistics.
- Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2021. [Benchmarking Natural Language Understanding Services for Building Conversational Agents](#), pages 165–183. Springer Singapore, Singapore.
- 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](#).
- Yosi Mass, Boaz Carmeli, Haggai Roitman, and David Konopnicki. 2020. [Unsupervised FAQ retrieval with question generation and BERT](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 807–812, Online. Association for Computational Linguistics.
- S. Mehri, M. Eric, and D. Hakkani-Tur. 2020. [Dialogue: A natural language understanding benchmark for task-oriented dialogue](#). *ArXiv*, abs/2009.13570.
- Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2021. [On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines](#). In *International Conference on Learning Representations*.

- Jason Phang, Thibault Févry, and Samuel R. Bowman. 2018. [Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks](#).
- 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.
- Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. [Recipes for building an open-domain chatbot](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online. Association for Computational Linguistics.
- Wataru Sakata, Tomohide Shibata, Ribeka Tanaka, and Sadao Kurohashi. 2019. [Faq retrieval using query-question similarity and bert-based query-answer relevance](#).
- Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. [Retrieval augmentation reduces hallucination in conversation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Noriko Tomuro and Steven Lytinen. 2004. Retrieval models and q and a learning with faq files. pages 183–202.
- Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. [Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval](#). *arXiv preprint arXiv:2112.07577*.
- Haode Zhang, Yuwei Zhang, Li-Ming Zhan, Jiaxin Chen, Guangyuan Shi, Xiao-Ming Wu, and Albert Y.S. Lam. 2021a. [Effectiveness of pre-training for few-shot intent classification](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1114–1120, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jianguo Zhang, Trung Bui, Seunghyun Yoon, Xiang Chen, Zhiwei Liu, Congying Xia, Quan Hung Tran, Walter Chang, and Philip Yu. 2021b. [Few-shot intent detection via contrastive pre-training and fine-tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1906–1912, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Jianguo Zhang, Kazuma Hashimoto, Wenhao Liu, Chien-Sheng Wu, Yao Wan, Philip Yu, Richard Socher, and Caiming Xiong. 2020. [Discriminative nearest neighbor few-shot intent detection by transferring natural language inference](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5064–5082, Online. Association for Computational Linguistics.

Appendix

A Dataset details

The details regarding different datasets is given in Table 7. The dataset BANKING77 consists of 77 intents for 1 domain, CLINC150 has 150 intents across 10 domains and HWU64 has 64 intents across 21 domains. Curekart, Powerplay11 and SOFMattress have 28, 59 and 21 intents respectively. Powerplay11 is a challenging set with some intents having only 1 sample even in full dataset and median samples per intent being 3 in the subset variant. The total number of Query-Question (Q-Q) pairs generated for each dataset is also shown.

Datasets	Intents (Domains)	Min	Max	Median	Samples	q-Q Pairs
BANKING77	77 (1)	30	167	108	8.6K	-
BANKING77-5	77 (1)	5	5	5	385	73920
BANKING77-10	77 (1)	10	10	10	770	296065
CLINC150	150 (10)	100	100	100	15K	-
CLINC150-5	150 (10)	5	5	5	750	280875
CLINC150-10	150 (10)	10	10	10	1500	1124250
HWU64	64 (21)	33	159	156	8.9K	-
HWU64-5	64 (21)	5	5	5	320	51040
HWU64-10	64 (21)	10	10	10	640	204480
Curekart	28 (1)	3	95	14	600	179700
Curekart-Subset	28 (1)	2	72	8	413	85078
SOFmattress	21 (1)	9	34	12	328	53628
SOFmattress-Subset	21 (1)	3	18	7	180	16110
Powerplay	59 (1)	1	46	7	471	110685
Powerplay-Subset	59 (1)	1	24	3	261	33930

Table 7: Details of datasets used along with number of Q-Q pair generated. Min, Max and Median are on samples per intent

Iterative Stratified Testing and Measurement for Automated Model Updates

Elizabeth Dekeyser*, Nicholas Comment, Shermin Pei, Rajat Kumar,†
Shruti Rai, Fengtao Wu, Lisa Haverty, Kanna Shimizu

Amazon Alexa

Abstract

Automating updates to machine learning systems is an important but understudied challenge in AutoML. The high model variance of many cutting-edge deep learning architectures means that retraining a model provides no guarantee of accurate inference on all sample types. To address this concern, we present Automated Data-Shape Stratified Model Updates (ADSMU), a novel framework that relies on iterative model building coupled with data-shape stratified model testing and improvement. Using ADSMU, we observed a 26% (relative) improvement in accuracy for new model use cases on a large-scale NLU system, compared to a naive (manually) retrained baseline and current cutting-edge methods.

1 Introduction

Research in automated machine learning, including Auto ML and Auto AI, has primarily focused on automating the steps between data preparation and creating a purpose-relevant machine learning model, which can include choosing model architecture, tuning hyperparameters, and data preprocessing (Truong et al., 2019; Feurer et al., 2020; He et al., 2021).

However, in production machine learning systems, there is an equally important task of automating the maintenance and updates of existing models currently in use. This task embodies two essential requirements: (i) sufficiently covering new use cases; and (ii) not regressing already working use cases.

In academic literature, there exist a range of methods to meet these requirements, including Bayesian approaches (Kirk, 2017; Theodoridis, 2015), mixture of experts models (Yuksel et al., 2012; Shazeer et al., 2017), finetuning (Käding et al., 2016; Finn et al., 2017; Yu et al., 2020), and

training data selection (Moore and Lewis, 2010; Axelrod et al., 2011; Iyer and Bilmes, 2013). However, production systems present unique challenges that are not addressed by these methods.

First, in deep learning models, high model variance can result in changing interpretations across model builds (Gal and Ghahramani, 2016; Belkin et al., 2019; Pham et al., 2020). Because of this, simply retraining a model with new training data can result in uneven coverage across a new use case, i.e. failing performance requirements for a subset and, thus not providing sufficient coverage for the new use case. Furthermore, retraining the model for the new use case can shift formerly correct interpretations, i.e. failing performance requirements for previously existing use cases. A standard solution to this challenge in the industry is repeated manual retraining and failure analysis, requiring weeks of modeler work for a simple model update.

Second, experimentation in production systems is further hampered by the costs of changing model architectures. Production systems can be in use by hundreds of millions of customers and are supported by complex engineering systems. Changes to these systems require extensive testing of both inference and engineering performance. Because of this, introducing architectural solutions to challenges of model variance or data diversity can be prohibitively costly.

To address these challenges, we present Automated Data-Shape Stratified Model Updates (ADSMU). ADSMU builds and iteratively improves models to optimize toward data-shape stratified metrics. It can be applied on top of pre-existing model architectures in any production system, making it compatible with both legacy and cutting-edge model architectures and methods for model updating.

Through stratified optimization, *ADSMU* ensures that models cover a diverse range of the data space without requiring manual intervention. Auto-

*edekeyse@amazon.com

†kmardpl@amazon.com

mated model updates using this framework resulted in a 26% (relative) decrease in new user experiences error rate compared to alternate model update approaches (refer to Table 1). In addition, the adoption of this AutoML framework corresponded to a 94% timeline reduction in its application to selected model retraining for supporting new NLU use cases, compared to manual retraining.

In the following sections, we discuss related work, examine the details of ADSMU and how it can be applied to automated model updates, and discuss the limitations of ADSMU.

2 Related Work

AI automation can encompass all the steps of building an ML model, including problem definition, data collection, data cleaning, data coding, metric selection, algorithm selection, parameter optimization, post-processing, deployment, online evaluation, and debugging (Feurer et al., 2020; He et al., 2021). Nevertheless, these steps tend to focus on the importance of building a first, robust model, with the assumption that updating and retraining can be done using an identical pipeline and framework (Truong et al., 2019).

Research on optimized model updates, separate from AutoML, has focused on two primary fields: First, model architectures, such as finetuning (Käding et al., 2016; Finn et al., 2017; Yu et al., 2020) or Bayesian approaches (Kirk, 2017; Theodoridis, 2015), and second, training data selection, which encompasses work on submodular optimization (Moore and Lewis, 2010; Axelrod et al., 2011; Iyer and Bilmes, 2013), cross-entropy comparison (Moore and Lewis, 2010), selection by proxy (Coleman et al., 2019), and active learning (Cohn et al., 1996; Settles, 2009; Liu et al., 2021).

Many of these methods are premised on developing rich training data that performs well across feature spaces. However, despite this focus on data diversity, performance is tested, measured, and improved based on average performance across development test sets. This inherently biases models to perform well on the majority data type, allowing more challenging or rarer test cases (corresponding to newly launched user experiences) to fall through the cracks.

3 Automated Data-Shape Stratified Model Updating

ADSMU provides a framework for (1) automating data stratification, (2) using this stratification for model measurement, and (3) boosting performance in failing stratification groups using model iteration. We discuss an application of this method using an iterative approach to model retraining, which is compatible with any model architecture.

3.1 Data-shape Based Stratification

3.1.1 Stratification Methods

ADSMU requires the implementation of a stratification “rule” that is used throughout the model update process. This rule takes advantage of the labelled nature of training data and uses those labels to split data up meaningfully. This can be done using various stratification methods, including exact match, fuzzy match, or model-based.

Exact match stratification relies on holding one part of the data constant and stratifying based on other division areas. In the case of natural language understanding, this could rely on utterance label shapes. For instance, the utterances “{please: Other} {play: Other} {moana: VideoName}” and “{please: Other} {play: Other} {frozen: VideoName}” would have identical utterance shapes and therefore, in a simplistic exact-match setting, would be part of the same stratification group.

Fuzzy match stratification relies on similar manually-input heuristics but allows for more variation. In the case of natural language understanding, this could be through matching slot trails rather than labels, in which case “{play: Other} {moana: VideoName}” and “{please: Other} {play: Other} {frozen: VideoName}” could be part of the same utterance shape.¹

The final method of data stratification is model-based. This uses unsupervised strategies to group together training data in a meaningful way and ensures data is optimized within those groups. For instance, a clustering method could be used on top of sentence embeddings that would take into account both content and syntax, placing “please play Moana” and “might you play Frozen” in Cluster

¹Both of the above methods require modeler insight to determine principled stratification heuristics, but this should depend on the use case. In one use case, stratifying based on entity content could be more valuable (with “Moana” and “Frozen” being the meaningful inputs), while in others, the semantic shape might matter more.

A and “it would be fantastic if you played Moana” and “I would really appreciate if you played Frozen” in Cluster B due to different carrier phrase structures i.e. the model learns to differentiate between two possible classes of carrier phrases for slotting in a named entity recognition task. While this method has the benefit of relying less on labels and manual input, it has the potential to introduce more noise into the modeling process.

3.1.2 Measuring Stratified Performance

Once established, these stratification groups become the foundation for future modeling and optimization.

During model training, model performance is not judged by a single accuracy metric but rather as an average of the performance of each of the stratification groups. This ensures each stratification group is equally weighted and under-performance in a less common test set is not overlooked.

In other words, traditional approaches measure performance as $\bar{x} = \frac{1}{n}(\sum_{i=1}^n x_i)$ where n is the number of test cases and x_i is a binary or continuous success metric for observation i .

However, we propose a measurement method that equally weights each stratification group, regardless of the number of observations within it:

$$\bar{x} = \frac{1}{L} \left(\sum_{h=1}^L \bar{x}_h \right) \quad (1)$$

where L is the number of stratum and \bar{x}_h is the average of the binary or continuous success metric for stratification group h .²

3.1.3 Resolving Stratification Group Failures

The final area in which stratification is applied is failure resolution. When a model shows failures in a stratification group, it adds synthetically-generated training data to improve performance in that stratification group.

This data is generated based on the user-defined heuristic for stratification. For instance, in an exact match example where the semantic shape is “play|Other TOKEN|VideoName” the system can generate fill-in-the-blank data for VideoName, either resampling from pre-existing training data (e.g. finding all VideoName instances and using that as

²There is a range of potential extensions to these measurement metrics. For instance, semantic groups could be weighted based on modeler-defined heuristics, such as predicted popularity. Alternately, other metrics, such as harmonic mean, could be used to calculate semantic group performance.

a catalog) or alternately using a user-provided catalog for data regeneration.³

3.2 Integration into Automated Model Updates

In this section, we discuss one method for integrating stratification groups into an end-to-end model update pipeline, following the steps indicated in Figure 1 and addressed in more detail below.

- 1. The user provides updated data.** The user inputs data to be added and iterated on top of a pre-existing model and dataset.
- 2. The system builds a stratified synthetic data pool.** The AI system ingests the user-provided data and splits it using a pre-defined stratification method (see Section 3.1.1).⁴
- 3. Training data addition and model building.** An initial amount of training data is added.⁵ Model is then trained agnostic to architectures, hyperparameters, and loss functions.
- 4. Results of model training evaluated.** The model results are evaluated on each of the stratification groups. Note that while the system may have added varying amounts of training data from different stratification groups, every stratification group is tested, and test performance is weighted equally.
- 5. Data addition for supporting failing stratification groups.** If a stratification group is underperforming, more data is added to support this failing shape using methods discussed in Section 3.1.3.

Steps 3-5 are then repeated until net performance, measured by the methods discussed in Section 3.1.2, reaches an acceptable level or, alternately, a preset maximum number of iterations is reached.

3.3 Experimental Setup

We examine the impact of the ADSMU framework applied to a machine learning task that mimics a production-scale system and compare its results with other standard model update approaches.

³This data could also be created using cutting-edge, model-based synthetic generation methods (Wan et al., 2017; Zhang et al., 2019). As with model-generated semantic groups, the additional value of these methods must be balanced against the noise and lack of precision introduced by model-generated data.

⁴This step can also include data generation or supplementation to increase the data size if necessary.

⁵This can be naive (equal amounts across stratification groups) or more principled (using model-based training data selection methods, e.g. Moore and Lewis (2010); Axelrod et al. (2011); Iyer and Bilmes (2013)).

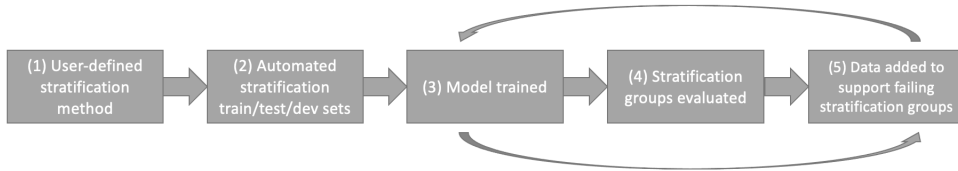


Figure 1: Exemplar integration of ADSMU in a model retraining pipeline. Data shape stratification is used for model performance measurement and improvement. Steps 3-5 are performed in iteration to boost model performance on weak stratification groups.

Experiment	Data Selection	Model Training	Overall Accuracy Change	New Use Case Accuracy Change
(1, Baseline)	Naive	Naive	0%	0%
(2)	Naive	Finetuning	1%	-2%
(3)	Submodular	Naive	0.6%	0.1%
(4)	Submodular	Finetuning	1.5%	-0.5%
(5)	ADSMU	ADSMU	1%	26%
(6)	ADSMU + Submodular	ADSMU	1.5%	28%

Table 1: Results from experimentation on experimental domain, intent, and named entity recognition tasks. Results reported are overall accuracy across classification tasks based on percent improvement from experiment (1, Baseline). ADSMU results used three model iterations.

Our experiments focused on updating a two-stage model for domain classification (DC) and a joint intent classification (IC) and named entity recognition (NER) task. For conducting experiments, we chose a deep bidirectional language model architecture based on their established success on several NLP tasks (Peters et al., 2018). The pre-existing training data (user data + synthetic data; user data used was already preprocessed for de-identification) proxied that of a large and complex production system and represented utterances across multiple domains and intents. We compare ADSMU to other model update approaches (Table 1) using a single model update task, wherein the model update was performed in a single pre-existing domain and introduced one new intent and new slots. We used exact match stratification to create 142 stratification groups for the training data and measured performance using the method from Equation (1). We set a goal of 95% accuracy for stratification groups, meaning that more training data was added only when groups did not meet that bar. This accuracy threshold is an experimental variable and can be tuned based on the trade-off between performance requirements and training data volume because error falls off as a power of the training data volume (Sorscher et al., 2022). Stratification group failures were generated using the exact match heuristic resampling from a user-provided catalog. All model update approaches used comparable amounts of training data. For pur-

poses of scale estimation, ADSMU started with training data measuring 0.004% of the pre-existing training data (in-domain and out-of-domain data), with each iteration (step 3-5 in Figure 2) adding more data for the failing stratification groups. The total utterances used during end-to-end ADSMU measured 0.008% of all the pre-existing training data.

We compared ADSMU to a naive retrained baseline and two common model refresh methods— submodular optimization and finetuning. The baseline model update was performed by an expert modeler by manual training data sampling to support the new intent in a pre-existing domain. To implement submodular optimization, we used methods presented in Schreiber et al. (2020); specifically, the implementation of the feature-based optimization method of Wei et al. (2014). This is based on a generalized feature-based function:

$$f(X) = \sum_{d=1}^D \phi(\sum_{i=1}^N X_{i,d}) \quad (2)$$

For the optimizer, we used an approximate lazy greedy algorithm (Schreiber et al., 2020). We added the same amount of training data for the naive baseline, ADSMU and submodular optimization. For comparison with finetuning, we froze the weights of the encoder layers and retrained the decoder to learn the new intent category. We retrained for 35% of the epochs of the baseline/ADSMU model.

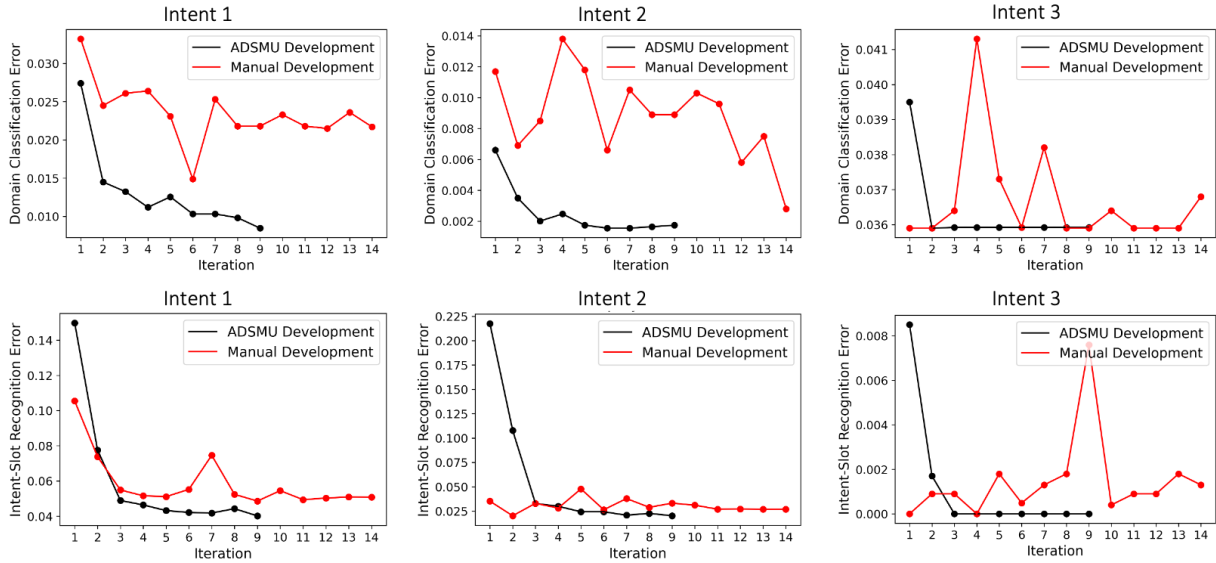


Figure 2: Comparison of manual model update vs. ADSMU. We obtained best model for the manual update after 14 iterations, and after 9 iterations for the ADSMU. Each manual iteration involved in-depth failure analysis and data addition for failure resolution by an expert modeler. Figure shows error rate (range: 0 to 1).

3.4 Results

For experiments in section 3.3, we report results based on percent improvement from current performance (baseline) based on overall accuracy across the three classification tasks (domain, intent, named entity recognition). While we only compared ADSMU to other approaches using a single model update, we repeated use of ADSMU for several other model building exercises that mimicked large-scale NLU production systems and obtained similar robust results. The results can be found in Table 1.

We find that finetuning performs slightly better overall than a naive retrained baseline but worse on new use cases. Submodular optimization combined with finetuning presents similar underperformance issues on new model use cases, though it slightly outperforms baseline when combined with a full retraining. ADSMU shows marked improvements in new use case accuracy compared with a naive baseline and slight improvements in overall accuracy. ADSMU combined with submodular optimization shows even more significant gains on overall and new use case accuracy, highlighting the potential to combine ADSMU with other data selection and model retraining approaches.

Additionally, we provide a specific exemplar case, wherein we compared iteration-level performance of model update performed by an expert modeler possessing domain knowledge against the

ADSMU for a spoken language understanding use case (Sarikaya et al., 2016) (same model and task as in section 3.3). This model update was performed in a single pre-existing domain and across multiple pre-existing intents but with new slots (Figure 2). The stop-criteria (best model) for the manual development work and the ADSMU iterations required: a) no regression of the pre-existing use cases; and b) less than 5% error rate for domain classification and intent and slot recognition for the new use case. To support the new use case, the expert modeler added training data measuring 0.5% in total of the pre-existing in-domain training data, and the ADSMU added data measuring 0.83% for obtaining the best model. The ADSMU converged faster than the manual model update process and performed better for both the domain classification and joint intent and named entity recognition task. Overall, the best model for ADSMU performed 0.8% better than the model provided by the expert modeler. ADSMU showed a relatively 24% lower error rate than the modeler model for the new use case. The iteration-level error rates (range: 0-1) for the new use case (across various intents) are shown in Figure 2.

The key to contextualizing these results is an analysis on a second axis – modeler cost. Accuracy results are compared against a single naive model rebuild baseline. Yet in practice, modelers can invest time to improve model accuracy through multiple rebuilds, offline testing, and tweaking training

data. Comparison of ADSMU to manual development represented a 94% timeline decrease in model updates from months to days.

4 Conclusions and Next Steps

ADSMU presents a framework for using stratified testing and measurement to iteratively improve models in the setting of automated model updates. This highlights the need that focuses on automating model creation while largely ignoring the challenges associated with automated model refresh processes (Truong et al., 2019; Feurer et al., 2020; He et al., 2021).

One key extension of this approach is applications that remove the need for model iteration and improve performance within stratified sub-groups in a single model build. For instance, optimizing on stratified loss functions might help ensure that models cover diverse use cases in a more consistent manner. More simplistically, a model could have training data added incrementally in between epochs based on weaker performing stratification groups.

5 Limitations

There are three key limitations to this approach. First, the reliance on user-defined stratification groups requires a certain degree of domain knowledge. Second, this framework is best suited to use with high-variance model architectures, since these are the most likely to show variable performance across data types and model builds. Finally, the reliance on iterations imposes a higher computational cost than a single model rebuild.

References

- Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. [Domain adaptation via pseudo in-domain data selection](#). In *Proceedings of the 2011 conference on empirical methods in natural language processing*, pages 355–362.
- Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. 2019. [Reconciling modern machine-learning practice and the classical bias–variance trade-off](#). *Proceedings of the National Academy of Sciences*, 116(32):15849–15854.
- David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. [Active learning with statistical models](#). *Journal of artificial intelligence research*, 4:129–145.
- Cody Coleman, Christopher Yeh, Stephen Musmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2019. [Selection via proxy: Efficient data selection for deep learning](#). *arXiv preprint arXiv:1906.11829*.
- Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. 2020. [Auto-sklearn 2.0: Hands-free automl via meta-learning](#). *arXiv preprint arXiv:2007.04074*.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. [Model-agnostic meta-learning for fast adaptation of deep networks](#). In *International conference on machine learning*, pages 1126–1135. PMLR.
- Yarin Gal and Zoubin Ghahramani. 2016. [Dropout as a bayesian approximation: Representing model uncertainty in deep learning](#). In *international conference on machine learning*, pages 1050–1059. PMLR.
- Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. [Automl: A survey of the state-of-the-art](#). *Knowledge-Based Systems*, 212:106622.
- Rishabh K Iyer and Jeff A Bilmes. 2013. [Submodular optimization with submodular cover and submodular knapsack constraints](#). *Advances in neural information processing systems*, 26.
- Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. 2016. [Fine-tuning deep neural networks in continuous learning scenarios](#). In *Asian Conference on Computer Vision*, pages 588–605. Springer.
- Matthew Kirk. 2017. *Thoughtful machine learning with Python: a test-driven approach*. " O'Reilly Media, Inc."
- Zhuoming Liu, Hao Ding, Huaping Zhong, Weijia Li, Jifeng Dai, and Conghui He. 2021. [Influence selection for active learning](#). In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9274–9283.
- Robert C Moore and William Lewis. 2010. [Intelligent selection of language model training data](#). In *Proceedings of the ACL 2010 conference short papers*, pages 220–224.
- Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. [Deep contextualized word representations](#). *CoRR*, abs/1802.05365.
- Hung Viet Pham, Shangshu Qian, Jiannan Wang, Thibaud Lutellier, Jonathan Rosenthal, Lin Tan, Yao-liang Yu, and Nachiappan Nagappan. 2020. [Problems and opportunities in training deep learning software systems: An analysis of variance](#). In *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, pages 771–783.
- R. Sarikaya, P. A. Crook, A. Marin, M. Jeong, J.P. Robichaud, A. Celikyilmaz, Y.B. Kim, A. Rochette, O. Z. Khan, X. Liu, D. Boies, T. Anastasakos, Z. Feizollahi, N. Ramesh, H. Suzuki, R. Holenstein, E. Krawczyk, and V. Radostev. 2016. [An overview of end-to-end language understanding and dialog management for personal digital assistants](#). pages 391–397.
- Jacob M Schreiber, Jeffrey A Bilmes, and William Stafford Noble. 2020. [apricot: Submodular selection for data summarization in python](#). *J. Mach. Learn. Res.*, 21:161–1.
- Burr Settles. 2009. [Active learning literature survey](#).
- 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*.
- Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2022. [Beyond neural scaling laws: beating power law scaling via data pruning](#). *arXiv preprint arXiv:2206.14486v3*.
- Sergios Theodoridis. 2015. *Machine learning: a Bayesian and optimization perspective*. Academic press.
- Anh Truong, Austin Walters, Jeremy Goodsitt, Keegan Hines, C Bayan Bruss, and Reza Farivar. 2019. [Towards automated machine learning: Evaluation and comparison of automl approaches and tools](#). In *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*, pages 1471–1479. IEEE.
- Zhiqiang Wan, Yazhou Zhang, and Haibo He. 2017. [Variational autoencoder based synthetic data generation for imbalanced learning](#). In *2017 IEEE symposium series on computational intelligence (SSCI)*, pages 1–7. IEEE.
- Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. 2014. [Submodular subset selection for large-scale speech training data](#). In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2020. [Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach](#). *arXiv preprint arXiv:2010.07835*.

Seniha Esen Yuksel, Joseph N Wilson, and Paul D Gader. 2012. [Twenty years of mixture of experts](#). *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. [Bertscore: Evaluating text generation with bert](#). *arXiv preprint arXiv:1904.09675*.

SLATE: A Sequence Labeling Approach for Task Extraction from Free-form Inked Content

Apurva Gandhi^{1*}, Ryan Serrao², Biyi Fang¹,
Gilbert Antonius¹, Jenna Hong¹, Tra My Nguyen³, Sheng Yi⁴,
Ehi Nosakhare¹, Irene Shaffer¹, Soundararajan Srinivasan¹, Vivek Gupta⁵
Microsoft
{¹firstname.lastname, ²ryserrao, ⁴shengyi, ³nguyenm, ⁵vivgupt}@microsoft.com

Abstract

We present SLATE, a sequence labeling approach for extracting tasks from free-form content such as digitally handwritten (or "inked") notes on a virtual whiteboard. Our approach allows us to create a single, low-latency model to simultaneously perform sentence segmentation and classification of these sentences into task/non-task sentences. SLATE greatly outperforms a baseline two-model (sentence segmentation followed by classification model) approach, achieving a task F1 score of 84.4%, a sentence segmentation (boundary similarity) score of 88.4% and three times lower latency compared to the baseline. Furthermore, we provide insights into tackling challenges of performing NLP on the inking domain. We release both our code and dataset for this novel task.

1 Introduction

The shift to remote and hybrid working styles due to COVID-19 has led to a large increase in virtual meetings. It has become increasingly important for participants to express themselves and brainstorm as naturally and effortlessly as possible, leading to an opportunity to extract entities from the large amounts of content created in these meetings. A natural entity of interest is a task created by a participant during the meeting which can be assigned to an individual to complete afterwards.

While past works have investigated extracting tasks from typed content such as emails (Bennett and Carbonell, 2005; Wang et al., 2019), there has been less focus on task extraction from more *free-form* content such as digitally handwritten (or *inked*) content on a virtual whiteboard or spoken content in a meeting. Extracting tasks from free-form content is challenging as it is often not as well-structured (e.g., poor grammar, inconsistent/lack of punctuation, typos, etc.) Furthermore, since this content first needs to be converted to text (e.g.,

*Corresponding author.

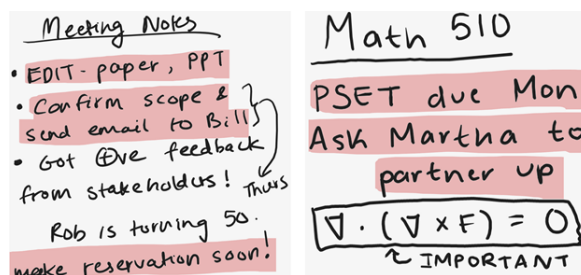


Figure 1: Examples of inked content. Task sentences are highlighted in red. Note the free-form style of the content, containing lists, a paragraph and annotations.

through a handwriting recognition model for ink or ASR for speech), downstream NLP models must be robust to errors made by the recognition models.

Moreover, as we discuss in Section 2, past approaches for task extraction from typed content assume text to already be segmented into sentences and focus on building a separate sentence-level task classification model. We cannot make this assumption for our scenario since automatic sentence segmentation models trained on typed content are unlikely to generalize well to the inconsistent capitalization and punctuation in free-form content (Stevenson and Gaizauskas, 2000; Rehbein et al., 2020). Furthermore, separating classification and sentence segmentation creates a latency-challenge: Classifying each sentence separately can cause the latency on CPU to scale linearly with the number of sentences, making real-time extraction challenging.

In our work, we address these challenges with our proposed approach SLATE – A Sequence Labeling Approach to Task Extraction from free-form content. Particularly, we apply this approach to inked content, such as that in Fig. 1, and call this application as Ink-SLATE. Our contributions are:

- We create a single, low-latency sequence labeling model to simultaneously perform sentence segmentation and task sentence classification for inked content in a document.
- We leverage ink document layout features to overcome inking domain challenges such as

the lack of punctuation and capitalization.

- We discuss a custom evaluation procedure suitable for the problem of extracting task sentences from free-form content and benchmark our SLATE approach against a baseline two-model approach.
- We compile a novel dataset for task extraction from ink document text. We release both our code and dataset in our linked repository.¹

2 Related Work

Past work on analyzing digitally inked content has dominantly fallen into either the category of handwriting recognition (Keysers et al., 2016; Gericke et al., 2012) or document layout analysis – grouping words into document lines (Ye et al., 2005), grouping document lines into blocks of spatially related text (Ye et al., 2007), or detecting indentation (Ye and Viola, 2004). Our work differs from these past works as it analyzes the semantics of the inked content rather than the layout. Furthermore, in an ink-analysis pipeline our work would sit downstream to handwriting recognition and layout analysis rather than replace them; we use, as input to our task extraction system, both recognized text and layout information extracted from the inked document. To our knowledge, our work is the first to tackle both sentence segmentation and task extraction for inked content.

Past work on task extraction has mainly been applied to typed content such as emails (Bennett and Carbonell, 2005; Wang et al., 2019). Other than our different domain of *inked content*, our work also differs in approach from these past works. Particularly, these works assume input text to be already segmented into sentences and focus only on building a classification model to classify these sentences into tasks/non-tasks. Thus, implicitly the task extraction systems in these works rely on a two-model approach: A sentence segmentation model to produce sentence candidates, followed by a classification model to classify sentences as tasks/non-tasks or different sub-categories of tasks. Our approach, on the other hand, uses a single model to simultaneously perform both sentence segmentation and classification for inked content.

Sequence labeling is an NLP approach that predicts a label for each token within a sequence,

rather than a label for the whole sequence. Applications of sequence labeling have traditionally included named entity recognition (NER), part-of-speech (POS) tagging, text segmentation, etc. In our work, we reframe the typically two-stage problem of task sentence extraction from documents as a single-stage sequence labeling problem. Since our approach uses a single, shared model to both segment and classify text, it can be thought of as a form of multi-task learning. Multi-task learning has shown to be data-efficient and less prone to overfitting to any single task (Crawshaw, 2020).

Most previous works on sequence labeling use Bi-LSTM and CRF layers in their models (He et al., 2020; Chen et al., 2020). We instead use a RoBERTa architecture, following the recent success of fine-tuning pretrained transformer LMs for data-constrained NLP. (Wolf et al., 2020).

3 Our Approach

3.1 Dataset

Our dataset consists of 200 vendor-created ink documents. To generate these documents, the vendors were provided various example templates with different content (to-do lists, recipes, brainstorm, general notes, etc.) that contain tasks and non-tasks written in various styles (single sentences, paragraphs, lists, diagrams, etc.) For additional diversity, the vendor employed 50+ different donors from different genders and age groups, and with various writing habits (e.g., left/right-handed).

After obtaining these ink documents, we passed them through a handwriting recognition engine (with 9.8% word error rate) and document layout analysis engine similar to ones referred to in Section 2 or to publicly available APIs (Apr, 2022). The result of this is 200 document text blocks and associated layout metadata (line breaks, bullets, etc.; see Section 3.2.4). The layout analysis also groups spatially related regions of text into separate blocks (similar to (Ye et al., 2007)), which we refer to as *writing regions*. We then split these writing regions between 6 annotators who performed two kinds of annotations: (1) Inserting sentence boundaries for sentence segmentation; (2) Annotating each sentence segment as a task/non-task.

Furthermore, the annotators also specially marked certain sentences which were only tasks or non-tasks in the context of the neighboring sentences. An example is a sentence with many mis-recognized words, making it incomprehensible in isolation; nevertheless, in the context of a to-do

¹Dataset, code, and additional details available at: <https://github.com/SLATEAuthors/SLATE>

list it may become apparent that the sentence is a task. Additional examples of tasks/non-tasks due to context can be found in Appendix A. Table 1 shows dataset split and annotation statistics. For annotation consistency, the annotators were provided a comprehensive annotation guide with example categories of sentences to be labeled as tasks and non-tasks. We release this guide in our linked repository. Since our task is novel, for reproducibility and to support future research, we also make the document texts, layout metadata, and task/sentence annotations available in our repository¹.

Next, we share domain-specific challenges:

Digital handwriting is often misrecognized: We rely on existing handwriting recognition models to convert handwritten strokes to text. Since handwriting is often messy and diverse, recognized text that is inputted to our task extraction model is often ridden with typos and non-sensical words.

Inked content is often overly concise: Users are generally not verbose when inking. Rather, they distill their content to important keywords, phrases, acronyms, phrases, etc. This concise style often lacks proper punctuation and grammar, making NLP tasks such as sentence segmentation to find the boundaries of task/non-task sentences quite challenging. For example, the first inked bullet in Fig. 1 lacks grammar, punctuation and is a list of keywords rather than a proper sentence. Furthermore, the lack of verbosity and the use of esoteric short-hands and acronyms make it even more challenging for a model to understand the meaning of the text. The third bullet in Fig. 1 shows such a shorthand – using ‘ \oplus ve’ instead of ‘positive.’

Ink users want to write in an unrestricted, free-form manner: Inking is conducive to brainstorming. Since people brainstorm tasks in various formats (to-do list, paragraphs, diagrams, mix of these styles, etc.), NLP systems built to analyze inked content must be able to handle this diversity. Fig. 1 shows an example of the free-form nature of inked content. Additional examples are in Appendix A.

3.2 Sequence Labeling Approach

At the heart of SLATE is sequence labeling. A sequence labeling approach treats the input document text as a sequence of tokens (or sub-words). It classifies each token as being part of one of a predefined set of classes. To extract our desired entities (e.g., sentences for sentence segmentation or task sentences for task extraction) we post-process

Content	Count	
	Train set	Test set
Ink documents	124	83
Sentences	2496	1416
Task sentences	704	440
Non-task sentences	1522	857
Task sentences due to context	173	54
Non-task sentences due to context	97	65

Table 1: Dataset statistics after annotation process.

the sequence of tokens according to their predicted class labels. A particular *sequence labeling scheme* determines the set of classes and the logic to post-process the predicted token-level class labels for entity extraction. In this work, we define and try three different sequence labeling schemes, described in the sections below. As will be discussed in Section 3.2.3, sequence labeling is key to letting us simultaneously perform sentence segmentation and task sentence classification with a single model.

3.2.1 Sentence-BI Labeling Scheme for Sentence Segmentation

The sentence-BI labeling scheme is used for sentence segmentation and is similar to schemes adopted by past works in sentence segmentation (Rehbein et al., 2020; Le, 2020). In this labeling scheme, tokens are assigned one of two labels: (B) - Beginning of Sentence; (I) - Inside of Sentence.

After the sequence labeling model classifies each token, we aggregate token-level class labels to word-level labels. This is done to make sure that we do not split sentences in the middle of words. The rule used for this aggregation is described in Algorithm 1 of Appendix B. Once we have predicted word-level labels, the words labeled as ‘B’ indicate the beginning of a new sentence, giving us the predicted sentence boundaries for sentence segmentation as shown in the top row of Fig. 2.

Since a model trained with the sentence-BI labeling scheme is only useful for sentence segmentation, for our task extraction scenario, we need an additional classification model to classify the segmented sentences into tasks/non-tasks. This two-model approach is precisely what we use as our baseline, described in Section 3.3.

3.2.2 SLATE-BIO Labeling Scheme for Task Extraction

The SLATE-BIO labeling scheme is used to extract *task* sentences from the input text. It assigns one

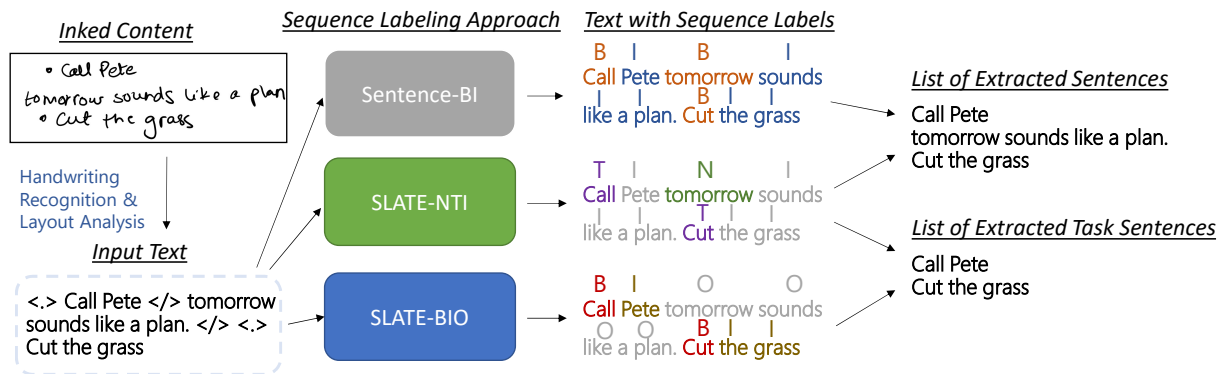


Figure 2: Illustration of the various sequence labeling configurations and how they are used to extract sentences and tasks from inked content.

of the following three labels to each token: (B) - Beginning of Task Sentence; (I) - Inside of Task Sentence; (O) - Outside of Task Sentence. BIO labeling schemes have commonly been used for NER and text chunking tasks (Sang and Buchholz, 2000; Ramshaw and Marcus, 1999). In our work, we adapt it for task sentence extraction.

Similar to sentence-BI, we aggregate predicted token-level labels to word-level labels (described in Algorithm 2 of Appendix B). Once we have the predicted word labels, a sequence of labels that starts with a ‘B’ and ends in zero or more ‘I’ labels indicates a task sentence. Task extraction with SLATE-BIO is illustrated at the bottom of Fig. 2.

3.2.3 SLATE-NTI Labeling Scheme for Task and Sentence Extraction

A disadvantage of SLATE-BIO is that while it finds the boundaries surrounding task sentences, it cannot be used for sentence segmentation as it does not find the boundaries between a contiguous block of non-task sentences. For this, we propose the SLATE-NTI labeling scheme to simultaneously train a sequence labeling model for both task sentence extraction and sentence segmentation (a form of multi-task learning). This scheme assigns one of the following three labels to each token: (N) - Beginning of a Non-Task sentence; (T) - Beginning of a Task Sentence; (I) - Inside of a Sentence.

Similar to the other schemes, we aggregate predicted token-level labels to word-level labels (Algorithm 3 of Appendix B). Once we have the predicted word labels, a sequence of word labels that starts with a ‘T’ and ends in zero or more ‘I’ labels indicates a task sentence, whereas a sequence that starts with an ‘N’ and ends in zero or more ‘I’ labels indicates a non-task sentence. Sentence segmentation and task sentence extraction using SLATE-NTI is illustrated in the middle row of Fig. 2.

3.2.4 Tackling Inking Peculiarities using Document Layout Metadata as Features

As mentioned in Section 3.1, inked content is often written in a casual style, lacking punctuation, proper grammar, verbosity, etc. Furthermore, upstream components like handwriting recognition can introduce errors. This makes modeling especially difficult as standard sentence segmentation relies on punctuation and capitalization to determine sentence boundaries. Similarly, misspelled words, acronyms and improper grammar make it difficult for a model to make sense of the sentence’s meaning and thus classify it. To compensate for these peculiarities of inked content, we supplement the input to our sequence labeling model with document layout metadata. Particularly, we add the following to our model input.

Line breaks indicate where a document line ends and a new one begins. While line breaks do not correspond exactly to sentence boundaries, we expect there to be strong correlation between their positions, providing a useful signal to the model for sentence segmentation purposes. We use the ‘</>’ token to indicate a line break in text.

Bullets are used to indicate the start of list items. People tend to write tasks in the form of to-do lists and thus it is common for tasks to be bulleted. Furthermore, bullets almost always indicate the start of a new sentence. Thus, we expect bullets to provide useful signal for both sentence segmentation and task classification. We use the ‘<./>’ token to indicate a bullet in text.

The left side of Fig. 2 shows how we add line breaks and bullets to the model input.

3.3 Baseline Approach

Our baseline approach is a two-model approach where we first train a sentence segmentation model

that takes as input the document text and outputs sentence boundaries. In our work, to build the sentence segmentation model, we use sequence labeling with the Sentence-BI labeling scheme as discussed in Section 3.2.1. We then train a separate sentence classification model that takes as input a sentence and outputs a task/non-task label.

3.4 Model Architecture

For each of the models we train, we fine-tune a pretrained RoBERTa (Liu et al., 2019) encoder implemented in the HuggingFace transformers library (Wolf et al., 2020). For the sequence labeling (SLATE-NTI, SLATE-BIO, and Sentence-BI) approaches, we add classification heads for each input token, to obtain the token-level sequence labels. For the classification model in the baseline approach, we use only a single classification head instead, for classifying one sentence at a time. Leveraging a pretrained transformer model allows us to obtain good performance even with our relatively small training set. For training and implementation details, please refer to Appendix C.

3.5 Evaluation Procedure

A challenge of evaluating SLATE is that since it performs sentence segmentation and classification *jointly*, it is ambiguous how to evaluate its task classification performance. Particularly, since the predicted sentence segments may not match the ground truth sentence segments, it is unclear how to compare their task/non-task labels. For example, consider Fig. 3. At the top, this figure shows sample input text for our task extraction system. The lower left shows the predicted annotation (sentence segmentation and task classification labels) while the lower right shows the ground truth annotation. The predicted task “send email & doc results” has words from two ground truth sentences – “send email & doc” and “results look great.” It is unclear which ground truth sentence we should compare its classification label with. Thus, without an explicit matching from predicted task segments to ground truth segments, it is ambiguous how to compare the labels of predicted and ground truth sentences. In our work, we use a bipartite graph matching algorithm to construct such an explicit matching using IOU similarity as edge weights between predicted and ground-truth sentences (Section 3.5.1). The result of this procedure is an explicit matching, allowing us to port typical classification metrics to our scenario. For example, Fig. 3 shows how we

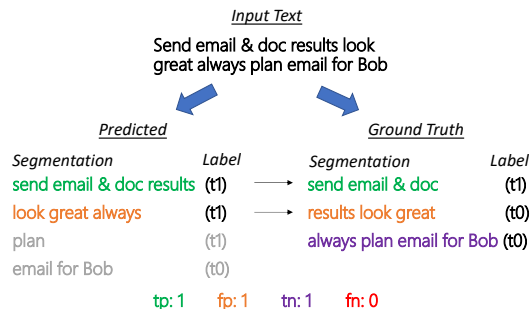


Figure 3: Example of calculating the number of true positives (tp), false positives (fp), true negatives (tn) and false negatives (fn) for the given input text, predictions and ground truth annotations. The t0/t1 labels are used as abbreviations for non-task/task labels.

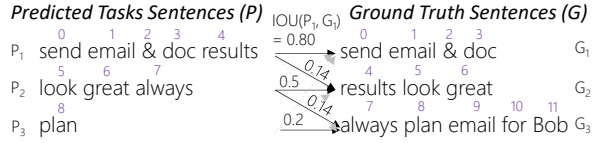
can calculate true/false positives/negatives, from which further classification metrics (e.g., accuracy, F1, etc.) can be computed. Additional discussion on our evaluation procedure and comparison to standard NER metrics is provided in Appendix D.

For evaluating the sentence segmentation performance of SLATE, we use the *boundary similarity (B)* sentence segmentation metric introduced in (Fournier, 2013). Concisely, B penalizes a predicted segmentation based on the number of edits required to transform the predicted segmentation to the ground truth segmentation. Near boundary misses are penalized less compared to full misses/additions. B is a score from 0-1 where a higher score represents a better predicted segmentation and a 1 represents a perfect segmentation. More metric details are in Appendix D.3.

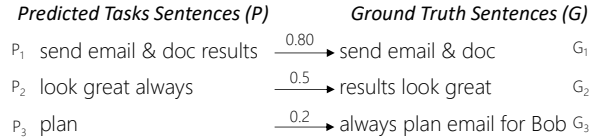
In our application, since the segmentation quality of extracted task sentences matters more than that of non-task sentences, we also compute a modified version of B which we call the *true positive boundary similarity (B_{tp})*. The formula to compute B_{tp} is the same as Equation 2 (Appendix D.3) except that in the segmentations that we compare, we only include the boundaries of true positive tasks.

3.5.1 Matching Predicted Task Sentences to Ground Truth Sentences

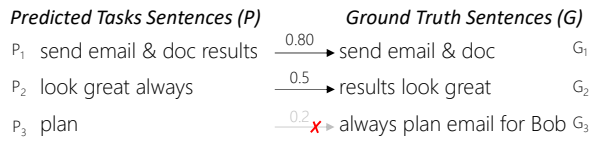
In this section, we describe the procedure used to obtain a matching between predicted task sentences and ground truth sentences. Let D represent the document text provided to the model to perform inference on. D is then a sequence of words w_i where $index(w_i)$ represents the positional index of w_i in D . Let G be a partition of D corresponding to the ground truth sentences (task and non-task) in D , i.e., each element $G_i \in G$ is a set of words



(a) Step 1 of the matching process constructs a complete weighted bipartite graph between the sets of predicted task sentences and ground truth sentences. The edge weights represent the IOU similarity between sentences, calculated on their respective sets of word indices (purple). The light gray edges have zero edge weights.



(b) Step 2 of the matching process finds the maximum weight full matching for the constructed graph.



(c) Step 3 of the matching process prunes out edges that do not meet a minimum similarity threshold.

Figure 4: Illustration of the procedure used to match predicted task sentences to ground truth sentences.

$w_j \in D$ representing a sentence in the ground truth segmentation of D . After performing inference with our model on D we observe the set P corresponding to the set of predicted task sentences, i.e, each element $P_i \in P$ is a set of words $w_j \in D$ representing a predicted task sentence. With this notation, we are now ready to describe the steps of the matching procedure:

1. **Construct a complete weighted bipartite graph between the sets P and G where each element (sentence) in the sets is a node and edge weights represent similarity between the node sentences.** A complete bipartite graph between sets P and G is one where every pair of nodes from differing sets have an edge but no pair of nodes from the same set has an edge. We use the Intersection over Union (IOU) between sentences in the graph to measure similarity. In our scenario, we define IOU as follows:²

$$IOU(P_i, G_j) = \frac{|P_{i-ind} \cap G_{j-ind}|}{|P_i \cup G_j|}$$

where $S_{-ind} := \{index(w_l) \mid w_l \in S\}$. (1)

²Using $P_{i-ind} \cap G_{j-ind}$ in the numerator of the definition of $IOU(P_i, G_j)$ instead of simply $P_i \cap G_j$ is important to avoid spurious matches when the same words may be repeated more than once in D .

2. **Find the maximum weight full matching M for the bipartite graph constructed above.**

We desire a matching between the sets P and G to maximize the overall similarity between matched sentences. Let $C = ((P, G), E)$ represent the weighted complete bipartite graph we constructed in first step, where the set E represents the set of edges in the graph and $weight(e)$ for $e \in E$ represents the IOU similarity score between the nodes that e connects. We construct a matching $M \subseteq E$ such that each node in the graph is included in at most one edge in M and so that $|M| = \min(|P|, |G|)$. When $|P| = |G|$, this is commonly known as a perfect matching. Since in our scenario we allow $|P|$ and $|G|$ to differ, we follow Karp (1980) and refer to this as a *full matching*. Furthermore, we choose the edges in M so as to maximize $\sum_{e \in M} weight(e)$, making M the maximum weight full matching for graph C . Essentially, M chooses the edges between predicted task sentences and ground truth sentences that maximizes the overall similarity between matched sentences. Also, note that the full matching allows at most one predicted segment to be matched to a ground truth sentence, preventing overcounting in the case where there are more than one predicted sentences for a single ground truth sentence.

3. **Prune M to remove matches that have non-significant overlap.** To provide credit only when predicted tasks are sufficiently similar to ground truth task sentences, we define a threshold t and remove edges $e \in M$ with $weight(e) < t$. In our work, we set $t = 0.25$.

Fig. 4 illustrates the matching procedure steps.

4 Results

The performance of the various modeling approach configurations are presented in Table 2. The first four rows show the performance of the different SLATE configurations tried and the last row shows the performance of the baseline approach. Our flagship approach is SLATE-NTI which has the following advantages: (1) The highest segmentation performance (B and B_{tp}); (2) Much lower latency compared to the baseline approach; (3) Better or at least comparable task classification performance with respect to the other approaches. The remainder of this section discusses some observed trends.

Method	Task (%)				Non-task (%)				Acc (%)	B_{tp} (%)	B (%)	Latency (ms)
	Rec	Prec	F1	Context Rec	Rec	Prec	F1	Context Rec				
SLATE-NTI with Doc Metadata	87.4	81.7	84.4	69.6	89.3	92.9	91.1	60.9	88.7	89.1	88.4	34.2
SLATE-BIO with Doc Metadata	88.9	80.4	84.4	75.6	88.3	93.6	90.8	64.0	88.5	86.6	-	
SLATE-NTI	90.2	81.2	85.5	78.1	88.6	94.4	91.4	59.1	89.2	84.3	83.4	26.5
SLATE-BIO	87.4	83.2	85.3	70.4	90.4	93.0	91.7	63.7	89.4	83.0	-	
Baseline	83.1	81.5	82.3	43.0	89.8	90.1	90.2	57.8	87.4	85.5	85.3	90.6

Table 2: Performance comparison of the various modeling approach configurations on our test set. The classification (recalls, precisions, F1s and accuracy) and segmentation metrics (B_{tp} and B) are averages over five distinctly seeded training runs for the corresponding modeling method. The latency values were obtained by first finding the mean latency of the model inference over each sample in the test set and then performing the average of five such runs.

4.1 The Latency Advantage of SLATE

As shown in Table 2, the SLATE approaches have 2.6 to 3.4 times lower inference latency compared to the the baseline approach, depending on whether they use document metadata or not. This lower latency of SLATE can be attributed to two main reasons. The first is that SLATE uses a single model for both segmentation and classification, whereas the baseline approach suffers from the combined latency of two separate models. The second reason is that since the classification model of the baseline approach acts on each sentence independently, its inference time scales linearly with the number of sentences in the input. The SLATE approach on the other hand can perform inference on an input containing multiple sentences in a single inference.

4.2 SLATE Benefits from Context

Unlike the sentence classification model in our baseline which has access to only a single sentence per inference, the SLATE approach has access to contextual information since it performs inference on a whole block of text at once. Table 2 shows the advantage that access to context gives SLATE compared to the baseline approach: Every SLATE approach has better classification performance (F1 scores and Accuracy) compared to the baseline. To further zoom in on the effect of contextual information, Table 2 also shows the recall of the approaches on sentences in the test set that were annotated as being tasks or non-tasks only due to context (see the Context Rec. columns). We see that each of the SLATE approaches has significantly higher recalls compared to the baseline on such sentences.

4.3 The Benefit of Multi-task Learning

SLATE with either the BIO or NTI labeling schemes is inherently a form of multi-task learning as it is trained to both segment text and classify

segmented text simultaneously. Still, SLATE-NTI has a stronger multi-task component compared to SLATE-BIO since unlike the BIO scheme, the NTI scheme forces the model to not only learn how to segment out task sentences but non-task sentences as well. Learning how to find the boundaries of non-task sentences is complementary to learning to find boundaries of task sentences. Thus, SLATE-NTI learns to be more effective at segmenting the text compared to SLATE-BIO. This can be seen in Table 2 by observing that the B_{tp} scores for SLATE-NTI configurations are higher than their corresponding SLATE-BIO configurations.

4.4 Adapting to Ink using Layout Metadata

As discussed in Section 3.2.4, we expect that adding document layout information such as line breaks and bullets to the model input should help compensate for the lack of traditional characteristics of natural language such as proper grammar, punctuation, capitalization, verbosity, etc. The results in Table 2 substantiate this expectation as we see large margins of improvement ($> 3.6\%$) in the segmentation metrics (B and B_{tp}) when we compare the SLATE approaches that use document metadata against those that do not. Thus, supplementing the model with document layout information is an effective method to adapt to the segmentation challenges of the inking domain.

5 Conclusion

We have presented SLATE, a single-model, sequence labeling approach for extracting tasks from free-form content. It overcomes ink domain challenges via our custom ink dataset and ink-document layout information. Our flagship configuration, SLATE-NTI, is a single, low-latency model trained for both accurate sentence segmentation and task sentence classification on inked content.

References

2022. Cross-platform handwriting recognition and interactive ink apis. <https://developer.myscript.com/>.
- Paul N Bennett and Jaime Carbonell. 2005. Detecting action-items in e-mail. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 585–586.
- Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020. Seqvat: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811.
- Nancy Chinchor and Beth Sundheim. 1993. **MUC-5 evaluation metrics**. In *Fifth Message Understanding Conference (MUC-5): Proceedings of a Conference Held in Baltimore, Maryland, August 25-27, 1993*.
- Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*.
- Chris Fournier. 2013. Evaluating text segmentation using boundary edit distance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1702–1712.
- Chris Fournier and Diana Inkpen. 2012. Segmentation similarity and agreement. *arXiv preprint arXiv:1204.2847*.
- Lutz Gericke, Matthias Wenzel, Raja Gumienny, Christian Willems, and Christoph Meinel. 2012. Handwriting recognition for a digital whiteboard collaboration platform. In *2012 International Conference on Collaboration Technologies and Systems (CTS)*, pages 226–233. IEEE.
- Zhiyong He, Zanbo Wang, Wei Wei, Shanshan Feng, Xianling Mao, and Sheng Jiang. 2020. A survey on recent advances in sequence labeling from deep learning models. *arXiv preprint arXiv:2011.06727*.
- Richard M Karp. 1980. An algorithm to solve the $m \times n$ assignment problem in expected time $o(mn \log n)$. *Networks*, 10(2):143–152.
- Daniel Keysers, Thomas Deselaers, Henry A. Rowley, Li-Lun Wang, and Victor Carbune. 2016. **Multi-language online handwriting recognition**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- The Anh Le. 2020. Sequence labeling approach to the task of sentence boundary detection. In *Proceedings of the 4th International Conference on Machine Learning and Soft Computing*, pages 144–148.
- 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*.
- Chunping Ma, Huafei Zheng, Pengjun Xie, Chen Li, Linlin Li, and Luo Si. 2018. Dm_nlp at semeval-2018 task 8: neural sequence labeling with linguistic features. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 707–711.
- 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.
- Ines Rehbein, Josef Ruppenhofer, and Thomas Schmidt. 2020. Improving sentence boundary detection for spoken language transcripts. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC), May 11-16, 2020, Palais du Pharo, Marseille, France*, pages 7102–7111. European Language Resources Association.
- Erik F Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. *arXiv preprint cs/0009008*.
- Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.
- Isabel Segura Bedmar, Paloma Martínez, and María Herrero Zazo. 2013. Semeval-2013 task 9: Extraction of drug-drug interactions from biomedical texts (ddiextraction 2013). Association for Computational Linguistics.
- Mark Stevenson and Robert Gaizauskas. 2000. Experiments on sentence boundary detection. In *Sixth Applied Natural Language Processing Conference*, pages 84–89.
- Julien Tourille, Matthieu Doutreligne, Olivier Ferret, Aurélie Névéol, Nicolas Paris, and Xavier Tannier. 2018. Evaluation of a sequence tagging tool for biomedical texts. In *proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 193–203.
- Wei Wang, Saghar Hosseini, Ahmed Hassan Awadallah, Paul N Bennett, and Chris Quirk. 2019. Context-aware intent identification in email conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 585–594.
- 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.

Ming Ye, Herry Sutanto, Sashi Raghupathy, Chengyang Li, and Michael Shilman. 2005. Grouping text lines in freeform handwritten notes. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 367–371. IEEE.

Ming Ye and Paul Viola. 2004. Learning to parse hierarchical lists and outlines using conditional random fields. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, pages 154–159. IEEE.

Ming Ye, Paul Viola, Sashi Raghupathy, Herry Sutanto, and Chengyang Li. 2007. Learning to group text lines and regions in freeform handwritten notes. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 1, pages 28–32. IEEE.

A Ink Document Examples

Here we provide additional examples of task/non-task sentences occurring in various styles in ink documents. Fig. 5 is an example of a to-do list style ink document. Here we see task sentences mainly written in the form of bullets some of which also span over multiple lines. Note certain sentences are tasks only based on the context and might not seem like task sentences otherwise.

Similarly, Fig. 6 shows an example of an inked recipe content. Although some of the sentences may seem like tasks, they are not in the context of being a recipe. For example, while “add tomato and garlic to make sauce” may be written like a task sentence, it is not considered a task sentence as it is an instruction in a recipe.

B Token-level to Word-level label Aggregation Rules for Sequence Labeling

Algorithm 1: Sentence-BI rule for aggregating token-level labels to word-level labels.

Input : $tokenLabels$ is a list of a token-level labels for a given word.

Output : Word-level label aggregated from $tokenLabels$.

```

1 if 'B' in tokenLabels then
2   | return 'B';
3 else
4   | return 'I';
5 end

```

Note capture assorted

- Print out MF Financials, } **Tasks**
- Return fan
- review agenda
- proof PPT; TRENDS etc
- confirm scope + fee to David Victor — email.
- YTD Performance
 - ↳ Market } PRINT OUT + REVIEW.
 - ↳ Firm
- Vulcan Graphics Bill resolved
- MF conclusions - review with JOE
 - By 3⁰⁰ PM ← **Multi-line tasks**
 - Finalize #'s 1⁰⁰
- RE+F
 - Lease issues w/ Mcormack
 - schedule next mtg with Lisa.
 - SF - Colliers; title search
- New Biz - phone
 - Network } **Tasks due to context**
 - Ester Bailey
 - Law Program - sh John Collier/Blast

Figure 5: Example of task sentences in an ink document.

- word brown meat w/ garlic & onion. } **Non-tasks due to context**
- add tomato and garlic to make sauce. Layer w/ cheese and lasagna noodles and bake for one hour @ 350°.
- needs more detail
- Non-task → healthy recipe

Figure 6: Example of non-task sentences in an ink document.

Algorithm 2: SLATE-BIO rule for aggregating token-level labels to word-level labels.

Input : $tokenLabels$ is a list of a token-level labels for a given word.

Output : Word-level label aggregated from $tokenLabels$.

```

1 if 'B' in tokenLabels then
2   | return 'B';
3 else
4   | return mode(tokenLabels);
5 end

```

Algorithm 3: SLATE-NTI rule for aggregating token-level labels to word-level labels.

Input : $tokenLabels$ is a list of a token-level labels for a given word.

Output : Word-level label aggregated from $tokenLabels$.

```
1 if 'N' or 'T' in tokenLabels then
2   | if # of 'T' labels > # of 'N' labels then
3   |   | return 'T';
4   | else
5   |   | return 'N';
6   | end
7 else
8   | return 'I';
9 end
```

C Training, Implementation and Hyperparameter Details

Each of our models were implemented using the HuggingFace Transformers library (Wolf et al., 2020). For sequence labeling models, we use the *AutoModelForTokenClassification* class with the *RoBERTa_{base}* architecture. For our sentence classification model in the baseline, we use the *AutoModelForSequenceClassification* class with the *RoBERTa_{base}* architecture. The model encoders were initialized using the pretrained weights provided by the library.

For training we use a batch size of 3 and 16 for the sequence labeling models and classification model respectively. All models were fine-tuned on our training set for 100 epochs. The objective for all the models was a class-weighted cross-entropy loss. The learning rate was kept constant at 1×10^{-6} for all of the models.

While training we use a machine with an NVIDIA RTX 2080ti GPU, Intel i9-9900K CPU and 64GB of RAM. For latency experiments, inference was done on the CPU of the above machine.

D Evaluation Procedure

D.1 Our Evaluation Approach

Our evaluation procedure has the following steps:

1. Match predicted task sentences to ground truth sentences (task and non-task) that have significant overlap. The matching procedure is described in more detail in Section 3.5.1.

2. Calculate the number of true/false positives/negatives according to the following definitions:

True Positive: Predicted task sentence that is matched to a ground truth task sentence.

False Positive: Predicted task sentence that is matched to a ground truth non-task sentence.

True Negative: Ground truth non-task sentence that is not matched to any predicted task sentence.

False Negative: Ground truth task sentence not matched to any predicted task sentence.

Fig. 3 shows the calculation of true/false positives/negatives for a sample input text, prediction and ground truth annotation.

3. Calculate standard classification metrics (accuracy, recall, precision, f1-scores, etc.) from the true/false positives/negatives to understand the task classification performance. When calculating non-task recall/precision, we treat true/false positives as true/false negatives and vice versa.

4. Calculate sentence segmentation metrics to evaluate the quality of the extracted task sentences. In this work we use the *boundary similarity (B)* sentence segmentation metric introduced in (Fournier, 2013). The boundary similarity metric is based on the *boundary edit distance (BED)* introduced in (Fournier and Inkpen, 2012). Concisely, the boundary similarity metric penalizes a predicted segmentation based on the number of edits required to transform the predicted segmentation to the ground truth segmentation. Near boundary misses are penalized less compared to full boundary misses/additions. B is a score from 0-1 where a higher score represents a better predicted segmentation and a 1 represents a perfect match to the ground truth segmentation. More metric details are in Appendix D.3.

In our application, since the segmentation quality of extracted task sentences matters more than that of non-task sentences, we also compute a modified version of this metric which we call the *true positive boundary similarity (B_{tp})*. The formula to compute B_{tp} is the same as Equation 2 (Appendix D.3) except that in the segmentations that we compare, we only include the boundaries of true positive tasks in the predicted segmentation and

the corresponding boundaries for the matched tasks in the ground truth segmentation.

D.2 Why not use standard NER Evaluation instead?

An alternative approach could be to leverage Named Entity Recognition (NER) metrics where the entities we are trying to recognize are task sentences. But these metrics are not without issues either. NER systems are typically evaluated by calculating precision, recall, and F1-scores at either the token level (Ma et al., 2018; Tourille et al., 2018) or at the entity level (Sang and De Meulder, 2003; Segura Bedmar et al., 2013). Token-level metrics suffer from being difficult to interpret compared to entity-level metrics. However, entity-level metrics are often too strict, giving credit only when predicted entities match the ground truth exactly (Sang and De Meulder, 2003). For example, in our scenario, if our model misses only a single token in an extracted task sentence, it would get no credit.

To address this, other evaluation schemes that provide credit for partial entity matches have been proposed (Segura Bedmar et al., 2013; Chinchor and Sundheim, 1993), but these tend to be more complex and difficult to interpret compared to the standard sentence classification metrics. In this work, we propose an evaluation procedure that allows us to calculate metrics that can be interpreted in the same way that standard sentence classification and segmentation metrics are, but at the same time provides enough slack to allow partial matches of predicted and ground truth task sentences.

D.3 The Boundary Similarity (B) Metric

Let us define a segmentation S of text to be a sequence of boundary positions where each boundary represents where a sentence begins and/or ends. A boundary can be placed between words and by default we place boundaries before the first word and after the last word in the text. Boundary edit distance (BED) is a measure of the minimum number of edits that need to be made to a given segmentation S_1 to make it identical to another segmentation S_2 . There are three types of edit operations we can make to S_1 in order to bring it into parity with S_2 :

- **Addition (A):** When S_1 is missing a boundary that is in S_2 we can add a boundary to S_1 .
- **Deletion (D):** When S_1 has a boundary where S_2 does not, we can delete this boundary from S_1 .

- **n -wise Transposition (T):** When S_1 misses a boundary in S_2 but has one in the near neighborhood, instead of making two edits to S_1 (one A and one D operation), we allow a single T operation which involves transposing/shifting the near boundary in S_1 to the corresponding position in S_2 . The parameter n determines how far boundaries in S_1 and S_2 can be to be considered for a T operation instead of an A or D operation. In this work, we set $n = 2$, allowing transpositions when boundaries differ by a maximum of 2 positions.

Suppose we calculate the BED for two segmentations of the same text S_1 and S_2 . The BED outputs the following: N_M is the number of perfect matches between boundaries, requiring no edits; N_A is the number of A operations required; N_D is the number of D operations required; the set $S_t = \{t \mid t = \# \text{ positions a boundary should be shifted} \forall T \text{ operations required}\}$. Then *boundary similarity* (B) between S_1 and S_2 is calculated as follows:

$$B(S_1, S_2) = 1 - \frac{N_A + N_D + \sum_{t \in S_t} \frac{t}{n}}{N_M + N_A + N_D + |S_t|} \quad (2)$$

Essentially, B gives no credit when a boundary is completely missed (A or D operation required) but gives partial credit when a near miss occurs (T operation required). For a more detailed explanation of this metric you may refer to (Fournier, 2013).

E Limitations

Here we discuss limitations of the work. First, since there are no past works and open datasets in the literature for our task, we are unable to benchmark against past works directly. To help address this, we have decided to open-source our dataset, modeling code, and evaluation code, so future research works can leverage these for benchmarking purposes. Still, this dataset is not very large. It consists of roughly 200 annotated ink documents. While this gave us a decent number of task and non-task sentences for fine-tuning a pretrained model and evaluating it on our task, with more data there are other approaches that we could take to further supplement our approach. For example, while we do try to address domain-specific noise such as errors introduced by handwriting recognition or poor grammar in inked content using document layout information, with a larger ink document corpus, we

could try supplementing our methodology for domain adaptation with language modeling. Finally, while we work with free-form content like inked documents, our work assumes the input to be recognized text from this content rather than the raw content (ink strokes). For example, handwriting recognition and document layout analysis methods are out of scope for this work. We cite examples of other works in literature and APIs that deal with these components in Sections 2 and 3.1.

Gaining Insights into Unrecognized User Utterances in Task-Oriented Dialog Systems

Ella Rabinovich

Matan Vetzler

David Boaz

Vineet Kumar

Gaurav Pandey

Ateret Anaby-Tavor

IBM Research

{ella.rabinovich1, matan.vetzler}@ibm.com

{vineeku6, gpandey1}@in.ibm.com

{davidbo, atereta}@il.ibm.com

Abstract

The rapidly growing market demand for automatic dialogue agents capable of goal-oriented behavior has caused many tech-industry leaders to invest considerable efforts into task-oriented dialog systems. The success of these systems is highly dependent on the accuracy of their intent identification – the process of deducing the goal or meaning of the user’s request and mapping it to one of the known intents for further processing. Gaining insights into unrecognized utterances – user requests the systems fail to attribute to a known intent – is therefore a key process in continuous improvement of goal-oriented dialog systems.

We present an end-to-end pipeline for processing unrecognized user utterances, deployed in a real-world, commercial task-oriented dialog system, including a specifically-tailored clustering algorithm, a novel approach to cluster representative extraction, and cluster naming. We evaluated the proposed components, demonstrating their benefits in the analysis of unrecognized user requests.

1 Introduction

The development of task-oriented dialog systems has gained much attention in both the academic and industrial communities over the past decade. Compared with open-domain dialog systems aimed at maximizing user engagement (Huang et al., 2020), task-oriented (also referred to as goal-oriented) dialog systems help customers accomplish a task in one or multiple domains (Chen et al., 2017). A typical pipeline system architecture is divided into several components, including a natural language understanding (NLU) module, which is responsible for classifying the first user request into potential *intents*, performing a decisive step that is required to drive the subsequent conversation with the virtual assistant in the right direction.

Goal-oriented dialog systems often fail to recognize the intent of natural language requests due

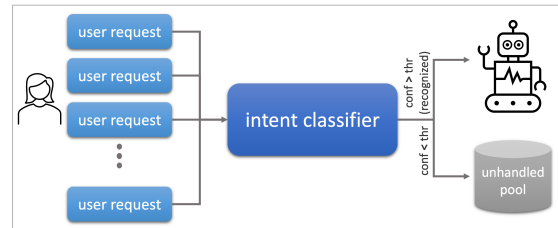


Figure 1: Natural language understanding (NLU) module. Based to the intent classifier’s confidence level, first user utterances are ‘recognized’ and associated with an execution flow, or stored in an unhandled pool.

to system errors, incomplete service coverage, or insufficient training (Grudin and Jacques, 2019; Kvale et al., 2019).¹ In practice, these cases are normally identified using intent classifier uncertainty. Here, user utterances that are predicted to have a level of confidence below a certain threshold to any of the predefined intents, are identified and reported as unrecognized or *unhandled*. Figure 1 presents the NLU module from a typical task-oriented dialog system: the user utterance is either transformed into an intent with an appropriate flow of subsequent actions, or labeled as unrecognized and stored in the *unhandled pool* (Figure 1).

Unhandled utterances often carry over various aspects of potential importance, including novel examples of existing intents, novel topics that may introduce a new intent, or seasonal topical peaks that should be monitored but not necessarily modeled within the system. In large deployments, the number of unhandled utterances can reach tens of thousands on a daily basis. Despite their evident importance for continuous bot improvement, tools for gaining effective insights into unhandled utterances have not been developed sufficiently, leaving a vast body of knowledge, as well as a range of

¹In most virtual assistants, a user utterance is considered unhandled by the system’s NLU module either by design (often referred to as “out-of-domain”), or due to the system’s failure to attribute the utterance to one of its existing intents.

potentially actionable items, unexploited.

Gaining insights into the topical distribution of unrecognized requests can be achieved using unsupervised text analysis tools, such as clustering or topic modeling. Indeed, identifying clusters of semantically similar utterances can surface topics of interest to a conversation analyst. We show that traditional clustering algorithms result in sub-optimal performance due to the unique traits of unhandled utterances in dialog systems: an unknown number of expected clusters and a very long tail of outliers. Consequently, we propose and evaluate a radius-based variant of the k-means clustering algorithm (Lloyd, 1982), that does not require a fixed number of clusters and tolerates outliers gracefully. We show that it outperforms its out-of-the-box counterparts on a range of real-world customer, as well as public datasets. The algorithm has recently been evaluated on the task of intent discovery in the context of large-scale, [production chatbot](#), being ranked first (out of 4) at coverage metrics, and second at utterance partitioning (Gretz et al., 2022).

We propose an end-to-end pipeline for surfacing topical clusters in unhandled user requests, including utterance cleanup, a designated clustering procedure and its extensive evaluation, a novel approach to cluster representatives extraction, and cluster naming. We approach this task in a real-world setting of commercial task-oriented dialog systems, and demonstrate the benefits of the suggested approach on multiple publicly available, as well as proprietary, datasets.

2 Clustering of Unrecognized Requests

Consider a virtual assistant aimed to attend to public questions about Covid-19. The rapidly evolving situation with the pandemic means that novel requests are likely to be introduced to the bot on a daily basis. As such, changes in international travel regulations would entail requests related to PCR test availability, and the decision to offer booster shots for seniors might cause a spike in questions about vaccine appointments for elderly citizens. Monitoring and prompt detection of these topics are fundamental for continuous bot improvement.

2.1 Clustering Utterances

Here we describe the main clustering procedure followed by an optional single merging step.

2.1.1 Main Clustering Procedure

Clustering requirements Multiple traits make up an effective clustering procedure in our scenario. First, the number of clusters is unknown, and has to be discovered by the clustering algorithm. Second, the nature of data typically implies several large and coherent clusters, where users repeatedly introduce very similar requests, and a very long tail of unique (often noisy) utterances that do not have similar counterparts. While the latter are of somewhat limited importance, they can amount to a significant ratio of the input data. There is an evident trade-off between the size of the generated clusters, their density or sparsity, and the number of outliers: smaller and denser clusters entail larger amounts of outliers. The decision regarding the precise outcome granularity may vary according to domain and bot maturity. Growing deployments, with a high volume of unrecognized requests, could benefit from surfacing large and coarse topics that are subject to automation. That said, mature deployments are likely to focus on fine-grained coherent clusters of utterances, introducing enhancements into the existing solution. Our third requirement is, therefore, a configurable density of the outcome clusters, which can be set up prior to the clustering procedure. Figure 2 illustrates a typical outcome of the clustering process; identified clusters are depicted in color, while the outliers, which constitute approximately half of the instances, appear in grey.

Existing clustering solutions can be roughly categorized across two major dimensions in terms of functional requirements: those requiring a fixed number of output clusters (1.a) and those that do not (1.b); those forcing cluster assignment on the entire dataset (2.a) and those tolerating outliers (2.b). Our clustering solution should accommodate (1.b) and (2.b): the number of clusters is determined by the clustering procedure, allowing for outliers. DBSCAN (Ester et al., 1996) and its descendant variants constitute a popular family of clustering solutions that satisfies these requirements; we, therefore, evaluate our algorithm against implementations of DBSCAN and its hierarchical version HDBSCAN (McInnes et al., 2017).

Data representation Given a set of m unhandled utterances $U=(u_1, u_2, \dots, u_m)$, we compute their vector representations $E=(e_1, e_2, \dots, e_m)$ using a sentence encoder. Multiple available encoders were evaluated for this purpose, considering both effectiveness and efficiency (see Section 2.2.1).



Figure 2: t-SNE projection of a sample of unrecognized user requests in a production task-oriented dialog system. Identified clusters are in color, outliers – in grey.

Radius-based clustering (RBC) We introduce a variant of the popular k-means clustering algorithm, complying with our clustering requirements by (1) imposing a strict cluster assignment criterion and (2) eventually omitting points that do not constitute clusters exceeding a predefined size.

Specifically, we iterate over randomly-ordered vectors in E , where each utterance vector can be assigned to an existing cluster if certain conditions are satisfied; otherwise, it initiates a new cluster. In order to join an existing cluster, the utterance is required to surpass a predefined similarity threshold min_sim with the cluster’s centroid,² implying its placement within a certain *radius* from the centroid. If multiple clusters satisfy the similarity requirement, the utterance is assigned to the cluster with the highest proximity i.e., the cluster with the highest semantic similarity to its centroid. Additional iterations over the utterances are further performed, re-assigning them to different clusters if needed, until convergence, or until a pre-defined number of iterations is exhausted.³ The amount of clusters generated by the final partition is controlled by the min_size value: elements that constitute clusters of small size (in particular, those with a single item) are considered outliers. Algorithm 1 presents the algorithm’s pseudo-code.

2.1.2 Cluster Merging

Cluster merging has been extensively used as a means to determine the optimal clustering outcome in the scenario where the ‘true’ number of

²Following the k-means notation, we compute a cluster’s centroid as the arithmetic mean of its member vectors.

³Contrary to k-means, our algorithm is not sensitive to its (random) initialization, since we are not required to select K centroids; utterance processing order has shown only negligible effect on the final outcome.

Algorithm 1: Radius-based Clustering

```

input:  $E$  ( $e_1, e_2, \dots, e_n$ ) /* elements */
input:  $min\_sim$  /* min similarity */
input:  $min\_size$  /* min cluster size */

 $C \leftarrow \emptyset$ 
while convergence criteria are not met do
  for each element  $e_i \in E$  do
    if the highest similarity of  $e_i$  to any existing
      cluster exceeds  $min\_sim$  then
      assign  $e_i$  to its most similar cluster  $c$ 
      re-calculate the centroid of  $c$ 
    else
      create a new cluster  $c'$  and assign  $e_i$  to it
      set the centroid of  $c'$  to be  $e_i$ 
      add  $c'$  to  $C$ 

  /*clusters with fewer elements than
    the predefined  $min\_size$  are
    considered outliers */

return: each  $c \in C$  of size exceeding  $min\_size$ 

```

partitions is unknown (Krishnapuram, 1994; Kaymak and Setnes, 2002; Xiong et al., 2004). These start with a large number of clusters and iteratively merge compatible partitions until the optimization criteria is satisfied. Beginning with fine-grained partitioning, we perform an (optional) single step of cluster merging, combining similar clusters into larger groups. A similar outcome could potentially be obtained by relaxing the min_sim similarity threshold and thereby, generating more heterogeneous flat clusters in the first place. However, a single step of cluster merging yielded results that outperform flat clustering on a range of datasets (see Table 3 and Section 2.2.2 for details). Classical agglomerative hierarchical clustering (AHC) algorithms merge pairs of lower-level clusters by minimizing the agglomerative criterion: a similarity requirement that has to be satisfied for a pair of clusters to be merged. Similar to AHC, we seek to merge clusters exhibiting high mutual similarity. In contrast to AHC, our approach is not pair-wise, rather it constitutes a subsequent invocation of the RBC that takes embeddings of the flat cluster centroids as its input.

Formally, given a set of clusters C of size $k=|C|$, identified by the algorithm, we compute the set of cluster centroid vectors $E^c=(e_1^c, e_2^c, \dots, e_k^c)$; these vectors are assumed to reliably represent the semantics of their corresponding clusters. E^c is further used as an input to subsequent invocation of the RBC algorithm, where the min_sim parameter can possibly differ from the previous invocation.

cluster name: difference covid flu (28)	cluster name: covid pregnancy (17)
is covid the same as the flu? (4)	covid 19 and pregnancy (10)
how is covid different from the flu? (3)	covid risks for a pregnant woman (4)
what is the difference between covid 19 and flu?	what is the risk of covid for pregnant women?
what’s the difference between covid and flu	is covid-19 dangerous when pregnant?
is the covid the same as cold?	7 months pregnant and tested positive for covid, any risks?
covid vs flu vs sars	covid 19 during pregnancy

Table 1: Example clusters of user requests generated by the RBC algorithm when applied on the Covid-19 dataset. Only a partial list of cluster members is presented in the table; the number in parenthesis denotes a cluster size.

Example Clustering Result Table 1 presents two example clusters generated from user requests to the Covid-19 bot. We applied the main RBC clustering procedure and a single subsequent merge step. Semantically related utterances are grouped together, where the number beside an utterance reflects its frequency in the cluster. As a concrete example, ‘is covid the same as the flu?’ was asked four times by different users.

2.2 Evaluation

We performed a comparative evaluation of the proposed clustering algorithm and HDBSCAN⁴, using common clustering evaluation metrics. The nature of the topical distribution of unrecognized utterances is probably most closely resembled by dialog systems *intent classification* datasets, where semantically similar training examples are grouped into classes, based on their intent. We used these classes to simulate cluster partitioning for the purpose of evaluation. We make use of three publicly available intent classification datasets (Liu et al. (2019), Larson et al. (2019) and Tepper et al. (2020)), as well as three datasets from real-world task-oriented chatbots in the domains of telecom, finance and retail. Table 2 presents the datasets details.

dataset	intents	examples	mean	STD
Liu et al. (2019)	46	20849	453.23	896.34
Larson et al. (2019)	150	22500	150.00	0.00
Tepper et al. (2020)	57	844	14.80	14.16
telecom	167	6364	38.10	26.74
finance	142	2301	16.20	25.28
retail	103	1714	16.64	11.42

Table 2: Datasets details: the number of intents, total training examples, mean and STD of the num of examples. We excluded out-of-scope examples from the Larson et al. (2019) dataset for the sake of evaluation.

⁴DBSCAN resulted in outcomes systematically inferior to HDBSCAN; hence, it was excluded from further experiments.

2.2.1 Evaluation Approach

The main approaches to clustering evaluation include extrinsic methods, which assume a ground truth, and intrinsic methods, which work in the absence of ground truth. Extrinsic techniques compare the clustering outcome to a human-generated *gold standard* partitioning. Intrinsic techniques assess the resulting clusters by measuring characteristics such as cohesion, separation, distortion, and likelihood (Pfitzner et al., 2009). We employ two popular extrinsic and intrinsic evaluation metrics: adjusted random index (ARI, (Hubert and Arabie, 1985)) and Silhouette Score (Rousseeuw, 1987). We vary the parameters of the RBC algorithm: merge type with none vs. single step (see Section 2.1.2); the encoder used for distance matrix construction: the SentenceTransformer (ST) encoder (Reimers and Gurevych, 2019) vs. the Universal Sentence Encoder (USE) (Cer et al., 2018); min similarity threshold used as a cluster “radius” was optimized on a held-out set of intents, per dataset. Both ARI and Silhouette yield values in the [-1, 1] range, where -1, 0 and 1 mean incorrect, arbitrary, and perfect assignment, respectively. The unique nature of our clustering requirements introduces a challenge to standard extrinsic evaluation techniques. Specifically, the min cluster size attribute controls the number of outliers, by considering only clusters that exceed the minimum number of members (see Figure 2). Aiming to mimic the ground truth partition (i.e, the intent classification datasets), we set the *min_size* attribute to the minimal class size in the dataset, subject to evaluation. As such, this attribute was set to 150 for the Larson et al. (2019) dataset, but to 2 for the finance dataset.

Both evaluation techniques assume full partitioning of the input space. Therefore, for our evaluation, we exclude the set outliers generated by our clustering algorithm altogether: only the subset of instances constructing the outcome clusters (e.g., instances depicted in color in Figure 2) was used to

compute both ARI and Silhouette. For completeness, we also report the ratio of a dataset utterances covered by the generated partition (‘% clst’ in Table 3), where the higher, the better.

2.2.2 Evaluation Results

Table 3 presents the results of our evaluation. Clearly, the RBC algorithm outperforms HDBSCAN across the board for both ARI and Silhouette scores, with the exception of the retail dataset, where the second best ARI score (0.37) is obtained by RBC along with over 80% of clustered utterances (compared to only 49.79% by HDBSCAN). HDBSCAN also outperforms RBC in terms of the ratio of clustered utterances for Liu et al. (2019) and the telecom dataset. However, these results are achieved by a nearly arbitrary partition of the input data, as mirrored by the extremely low ARI and Silhouette scores. We conclude that RBC outperforms its out-of-the-box counterpart on virtually all datasets in this work. The ratio of clustered examples (‘% clst’) exhibits considerable variance among the datasets; this result is indicative of the varying levels of semantic coherence of the underlying intent classes, which are typically constructed manually by a bot designer. As such, over 87% of all training examples were covered by the clustering procedure for the retail dataset, but only 33.90% for Larson et al. (2019).

The extremely poor results obtained for the telecom dataset by HDBSCAN stem from its clustering outcome that only contains two clusters: (1) a small group of unique examples and (2) all the rest.

Runtime and Memory Due to its nearly polynomial complexity, the proposed clustering algorithm may entail efficiency considerations for a very large amount of data. As such, with pre-computed request embeddings, clustering 20K unhandled requests results in less than 10 seconds, while clustering 85K requests takes 82 seconds with over 850MB of RAM consumption. All experiments were conducted on a server with 8 CPUs.

3 Selecting Cluster Representatives

Contemporary large-scale deployments of virtual assistants must cope with increasingly high volumes of incoming user requests. A typical large task-oriented system can accept over 100K requests (i.e., user utterances) per day, where the amount of conversations that pass the initial step of intent identification varies between 40% and 80%. Con-

		algo		RBC				HDBSCAN	
		merge type		no merge		single step		—	
		encoder		USE	ST	USE	ST	USE	ST
Liu	ARI	0.42	0.40	0.74	0.44	0.42	0.03		
	Silhouette	0.47	0.42	0.67	0.50	0.39	0.09		
	% clst	12.12	12.03	12.12	16.09	12.69	38.36		
Larson	ARI	0.89	0.86	0.68	0.76	0.49	0.69		
	Silhouette	0.47	0.50	0.48	0.50	0.39	0.47		
	% clst	16.29	32.60	16.29	33.90	24.92	32.98		
Tepper	ARI	0.66	0.65	0.73	0.52	0.69	0.67		
	Silhouette	0.45	0.49	0.51	0.37	0.45	0.46		
	% clst	79.68	85.12	79.68	88.18	58.31	60.15		
telecom	ARI	0.32	0.54	0.63	0.38	0.00	0.00		
	Silhouette	0.17	0.20	0.18	0.11	0.00	0.00		
	% clst	25.18	46.87	25.18	59.78	83.24	97.90		
finance	ARI	0.40	0.42	0.45	0.56	0.45	0.49		
	Silhouette	0.37	0.39	0.35	0.32	0.34	0.35		
	% clst	47.59	63.26	47.59	74.28	23.46	36.22		
retail	ARI	0.28	0.37	0.31	0.24	0.37	0.38		
	Silhouette	0.24	0.28	0.22	0.27	0.23	0.32		
	% clst	68.19	80.43	68.19	87.18	37.55	49.79		

Table 3: Clustering evaluation results; ‘% clst’ denotes the ratio of clustered examples out of total; the best result in a row is boldfaced.

sequently, tens of thousands of requests can be identified as unrecognized on a daily basis. Clustering these utterances would result in large clusters that are often impractical for manual processing. Providing conversation analysts with a limited set of *cluster representatives* is a fundamental step toward extracting value from the unrecognized data.

3.1 Representative Characteristics

A plausible set of representative cluster utterances has to satisfy two desirable properties: utterance *centrality* and *diversity*. We define an utterance centrality to be proportional to its frequency in a cluster: requests with higher frequency should be boosted, since they are typical of the way people express their needs to the bot. The diversity of the utterance set mirrors the subtle differences in the phrasing and meaning of utterances; these reflect the various ways people can express the same need.

Sampling randomly from a cluster may result in a sub-optimal set of representatives, in terms of both centrality and diversity. Consider the example where no ‘covid 19 and pregnancy’ requests (Table 1, right) are selected as representatives (low centrality), or both ‘what is the difference between covid 19 and flu?’ and ‘what’s the difference between covid and flu?’ (Table 1, left) are selected (low diversity). Contrary to these examples, the set {‘is covid the same as the flu?’, ‘is the covid the same as cold?’, ‘covid vs flue vs sars’} contains utterance of high centrality (the first utterance), and compre-

hensive coverage of the entire cluster semantics.

3.2 Selecting Representatives

To ensure diversity and centrality among the selected representatives, we use determinantal point process (DPP). Specifically, we consider a restricted class of DPPs known as L-ensembles. Given a set of items, \mathcal{S} , L-ensembles define a probability distribution over the power set of \mathcal{S} . Equivalently, L-ensembles define a probability distribution over binary vectors of length $|\mathcal{S}|$, where the i^{th} entry in the vector indicates if the i^{th} item in \mathcal{S} was included in the subset or not. These indicator variables are negatively correlated where the correlations are governed by a positive semidefinite matrix K . L-ensembles ensure that the more similar two items are, as indicated by the corresponding entry in the kernel matrix, the less likely are they to occur in the same sampled subset. Thus, it is an excellent model for ensuring diversity among the selected representatives.

Given a positive semi-definite kernel matrix K , the probability of $A \subset \mathcal{S}$ is governed in an L-ensemble as $P(A) \propto \det(K_A)$, where K_A is the restriction of K to the indices present in the subset A . We construct the kernel matrix to ensure that samples from the L-ensemble have high centrality while also being diverse. To achieve this, we first project the embeddings of the utterances within the cluster onto a unit sphere. We further take into consideration the factor of centrality by scaling the vectors' length based on their frequency in the cluster. Given the resultant embeddings E , where the embedding of the i^{th} entry is the i^{th} row vector in E , the kernel matrix is obtained by $K = EE^T$. Thus, the $(i, j)^{th}$ entry of the kernel corresponds to the angle between the i^{th} and j^{th} vector scaled by the frequency of occurrence of those vectors. We make use of the freely available [DPPy Python package](#) for sampling a subset of representatives, given the above kernel matrix.

Evaluation Using the clustering approach in Section 2 we extracted 50 clusters of varying sizes of unhandled user requests from a large-scale production system. A set of three cluster representatives was extracted using the technique described in this section, along with two baselines: (1) three random cluster members, (2) three unique most frequent cluster members. Three in-house annotators labeled their preferred alternative, satisfying *centrality* and *diversity* properties in the best way. The

majority vote was obtained in 47 out of 50 cases, with 37 out of 47 (79%) choices preferring the centrality-diversity approach. The mean pairwise Cohen's Kappa between the annotators was 0.44.

4 Cluster Naming

Assigning cluster with names, or labels, is an essential step toward their consumability. Common approaches to this task resort to simple but reliable techniques based on word n-gram extraction, such as *tf-idf*; many of these techniques made their way into the first large-scale information retrieval (IR) systems (Ramos et al., 2003; Aizawa, 2003). Here, we distinguish between the task of cluster naming (extracting a coherent phrase reliably reflecting a cluster's content) and the task of keyword extraction (providing a sequence of one or more words for a compact representation of a document).

Common approaches to cluster naming include extracting one of the cluster's members to reflect the cluster's content; extracting such a member can be done by naively selecting the most frequent member in the cluster or by choosing a member satisfying maximum cosine similarity to the cluster's centroid (Alicante et al., 2016). In other cases, a good name may not occur directly as one of the cluster's members, and hence requires different handling. Some works were trying to investigate the contribution of external knowledge-bases for cluster naming, by incorporating Wikipedia pages' meta-data corresponding to the cluster's content (Carmel et al., 2009), while others were trying to generate clusters' queries, as a mixture of cluster-internal and differential labeling (Hagen et al., 2015). Contemporary large pretrained large language models can also be used for the task of keyword extraction. Here we make use of KeyBERT – an approach based on BERT (Devlin et al.) – for identifying key phrases in a cluster, and evaluate the outcome against *tf-idf*.

Cluster Labeling with *tf-idf* We treat all utterances in individual clusters from a set $C = (c_1, c_2, \dots, c_k)$ as distinct documents. We first applied lemmatization to these documents using the [spacy toolkit](#) (Honnibal and Montani, 2017), excluded stopwords, and further ranked all ngram token sequences of length N (for $N \in (1, 2, 3)$) by their *tf-idf* score. The ngram with the highest score was selected as the cluster name.

Cluster Labeling with KeyBERT Treating each cluster as a document, we first extract document-level representation using a pretrained BERT language model.⁵ We further extract ngram representations for all unique word ngrams in the document, and compute semantic similarity between each ngram’s embedding and that of the document. Ngram with the highest cosine similarity to the document is selected as the cluster name.⁶

Evaluation Adhering to the same evaluation paradigm as Section 2.2, we use the six intent classification datasets for assessing the quality of cluster naming techniques. A common practice for building an intent training dataset involves assigning each class in the training set with a meaningful name, typically mirroring the semantics of the class. As such, an intent class grouping example requests about Covid-19 testing information in [Tepper et al. \(2020\)](#), is named ‘testing information’. For each class in the intent training set, we compare the automatically extracted class name to that assigned to the class by the dataset creator, where the similarity is obtained by encoding the two phrases – the original class name and the candidate one – and computing their cosine similarity.

Table 4 presents the results for the two methods. Neither approach systematically outperforms the other, and the only significant difference in favor of the tf-idf approach is found for [Liu et al. \(2019\)](#). We, therefore, conclude that the two approaches are roughly comparable and adhere to the faster tf-idf method in our pipeline solution.

dataset	tf-idf	KeyBERT
Liu et al. (2019)	0.718*	0.626
Larson et al. (2019)	0.555	0.489
Tepper et al. (2020)	0.481	0.460
telecom	0.437	0.470
finance	0.438	0.426
retail	0.375	0.393

Table 4: Cluster naming evaluation: for each dataset, the mean pairwise similarity between the predefined intent name and the assigned keyphrase is presented. ‘*’ denotes significant difference at p-val<0.01 using the Wilcoxon (Mann–Whitney) ranksums test.

5 Conclusions and Future Work

Analyzing unrecognized user requests is a fundamental step toward improving task-oriented dia-

⁵We use ‘all-MiniLM-L6-v2’ model in our experiments.

⁶We make use of the freely available [KeyBERT](#) package.

log systems. We present an end-to-end pipeline for clustering, representatives selection, and cluster naming – procedures that facilitate the effective and efficient exploration of utterances unrecognized by the NLU module. We propose a clustering variant of the popular k-means algorithm, and show that outperforms its out-of-the-box alternatives on multiple metrics. We also suggest a novel approach to extracting representative utterances while simultaneously optimizing their centrality and diversity.

Our future work includes the evaluation of our clustering approach with additional datasets, exploration of additional approaches to representative set selection, and advanced techniques for cluster naming. Leveraging clustering results to automatically identify actionable recommendations for conversation analyst is another venue of significant practical importance, we plan to pursue.

6 Ethical Considerations

Cluster representative sets (Section 3) were annotated by in-house workers who were compensated with above minimum wages. To protect user privacy, no personally identifiable information (e.g., name, address) were presented to the annotators.

Acknowledgements

We are grateful to the anonymous reviewers and the meta reviewer for their constructive feedback. We would also like to thank Chani Sacharen for her kind help with earlier versions of this work.

References

- Akiko Aizawa. 2003. [An Information-Theoretic Perspective of tf-idf Measures](#). *Information Processing & Management*, 39(1):45–65.
- Anita Alicante, Anna Corazza, Francesco Isgrò, and Stefano Silvestri. 2016. [Semantic cluster labeling for medical relations](#). In *International Conference on Innovation in Medicine and Healthcare*, pages 183–193. Springer.
- David Carmel, Haggai Roitman, and Naama Zwerdling. 2009. [Enhancing cluster labeling using wikipedia](#). In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146.
- Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. [Universal Sentence Encoder](#). *arXiv preprint arXiv:1803.11175*.

- Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. [A Survey on Dialogue Systems: Recent Advances and New Frontiers](#). *Acm Sigkdd Explorations Newsletter*, 19(2):25–35.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. [Bert: Pre-training of deep bidirectional transformers for language understanding](#). *arXiv preprint arXiv:1810.04805*.
- Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiawei Xu. 1996. [Density-Based Spatial Clustering of Applications with Noise](#). In *Int. Conf. Knowledge Discovery and Data Mining*, volume 240, page 6.
- Shai Gretz, Assaf Toledo, Roni Friedman, Dan Lahav, Rose Weeks, Naor Bar-Zeev, João Sedoc, Pooja Sangha, Yoav Katz, and Noam Slonim. 2022. [Benchmark data and evaluation framework for intent discovery around covid-19 vaccine hesitancy](#). *arXiv preprint arXiv:2205.11966*.
- Jonathan Grudin and Richard Jacques. 2019. [Chatbots, Humbots, and the Quest for Artificial General Intelligence](#). In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*.
- Matthias Hagen, Maximilian Michel, and Benno Stein. 2015. [What was the query? generating queries for document sets with applications in cluster labeling](#). In *International Conference on Applications of Natural Language to Information Systems*, pages 124–133. Springer.
- Matthew Honnibal and Ines Montani. 2017. [spaCy 2: Natural Language Understanding with Bloom Embeddings, Convolutional Neural Networks and Incremental Parsing](#). *Sentometrics Research*.
- Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. 2020. [Challenges in Building Intelligent Open-Domain Dialog Systems](#). *ACM Transactions on Information Systems (TOIS)*, 38(3):1–32.
- Lawrence Hubert and Phipps Arabie. 1985. [Comparing Partitions](#). *Journal of classification*, 2(1).
- Uzay Kaymak and Magne Setnes. 2002. [Fuzzy Clustering with Volume Prototypes and Adaptive Cluster Merging](#). *IEEE Transactions on Fuzzy Systems*, 10(6):705–712.
- Raghu Krishnapuram. 1994. [Generation of Membership Functions via Possibilistic Clustering](#). In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*, pages 902–908. IEEE.
- Knut Kvale, Olav Alexander Sell, Stig Hodnebrog, and Asbjørn Følstad. 2019. [Improving Conversations: Lessons Learnt from Manual Analysis of Chatbot Dialogues](#). In *International workshop on chatbot research and design*, pages 187–200. Springer.
- Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. [An Evaluation dataset for intent classification and out-of-scope prediction](#). 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 1311–1316.
- Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. [Benchmarking Natural Language Understanding Services for Building Conversational Agents](#). In *10th International Workshop on Spoken Dialogue Systems Technology 2019*, volume 714, pages 165–183. Springer.
- Stuart Lloyd. 1982. [Least Squares Quantization in PCM](#). *IEEE transactions on information theory*, 28(2):129–137.
- Leland McInnes, John Healy, and Steve Astels. 2017. [hdbscan: Hierarchical Density Based Clustering](#). *Journal of Open Source Software*, 2(11):205.
- Darius Pfitzner, Richard Leibbrandt, and David Powers. 2009. [Characterization and Evaluation of Similarity Measures for Pairs of Clusterings](#). *Knowledge and Information Systems*, 19(3):361–394.
- 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.
- 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.
- Peter J Rousseeuw. 1987. [Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis](#). *Journal of computational and applied mathematics*, 20:53–65.
- Naama Tepper, Esther Goldbraich, Naama Zwerdling, George Kour, Ateret Anaby Tavor, and Boaz Carmeli. 2020. [Balancing via Generation for Multi-Class Text Classification Improvement](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1440–1452.
- Xuejian Xiong, Kap Luk Chan, and Kian Lee Tan. 2004. [Similarity-Driven Cluster Merging Method for Unsupervised Fuzzy Clustering](#). In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 611–618.

CoCoID: Learning Contrastive Representations and Compact Clusters for Semi-Supervised Intent Discovery

Qian Cao¹, Deyi Xiong^{2,*}, Qinlong Wang¹ and Peng Xia¹

¹Leyan Tech, Shanghai, China

²College of Intelligence and Computing, Tianjin University, Tianjin, China

caoqian0905@gmail.com; dyxiong@tju.edu.cn;

qinlong.wang@leyantech.com; pengxia24@163.com

Abstract

Intent discovery is to mine new intents from user utterances, which are not present in the set of manually predefined intents. Previous approaches to intent discovery usually automatically cluster novel intents with prior knowledge from intent-labeled data in a semi-supervised way. In this paper, we focus on the discriminative user utterance representation learning and the compactness of the learned intent clusters. We propose a novel semi-supervised intent discovery framework CoCoID with two essential components: contrastive user utterance representation learning and intra-cluster knowledge distillation. The former attempts to detect similar and dissimilar intents from a minibatch-wise perspective. The latter regularizes the predictive distribution of the model over samples in a cluster-wise way. We conduct experiments on both real-life challenging datasets (i.e., CLINC and BANKING) that are curated to emulate the true environment of commercial/production systems and traditional datasets (i.e., StackOverflow and DBPedia) to evaluate the proposed CoCoID. Experiment results demonstrate that our model substantially outperforms state-of-the-art intent discovery models (12 baselines) by over 1.4 ACC and ARI points and 1.1 NMI points across the four datasets. Further analyses suggest that CoCoID is able to learn contrastive representations and compact clusters for intent discovery.

1 Introduction

Intent discovery is to pinpoint novel intents from user utterances, which are not present in the set of predefined intents. Discovering novel intents for goal-oriented dialogue has recently attracted growing attention and interest (Lin et al., 2020; Zhang et al., 2021), not only because it is difficult to manually define all potential user intents for conversational agents deployed in a wide range of

* Corresponding author

Model	NMI \uparrow	ARI \uparrow
sup-simcse-bert- large -uncased	80.25	66.93
sup-simcse-bert- base -uncased	75.09	59.52

Table 1: Comparison results of the DeepAligned model with supervised SimCSE-BERT_base vs. SimCSE-BERT_large on the test set of DBPedia.¹Normalized Mutual Information (NMI) and Adjusted Rand Index (ARI) are widely-used clustering evaluation metrics (see Section 5.1 for more details).

real-world scenarios, but also due to the cost of curating intent-labelled data.

Approaches to intent discovery usually include unsupervised clustering methods (Kathuria et al., 2010; Cheung and Li, 2012; Padmasundari and Bangalore, 2018) and semi-supervised methods (Basu et al., 2004; Hsu et al., 2018; Han et al., 2019). The former performs unsupervised clustering algorithms, e.g., K-means clustering, to group user utterances into clusters according to their underlying intents. The latter attempts to inject weakly supervised signals (Haponchyk et al., 2018; Caron et al., 2018), or external knowledge (Lin et al., 2020; Zhang et al., 2021) into the procedure of clustering. Compared with the unsupervised clustering methods, models that explore supervised signals or external knowledge are capable of discovering better user intents (Zhang et al., 2021). In this paper, we follow the semi-supervised research philosophy for intent discovery.

However, different from previous works (Lin et al., 2020; Zhang et al., 2021), our focus lies on two aspects: user utterance representation and the compactness of clusters. For utterance representation, we have conducted a simple investigation in our preliminary experiments. With the DeepAligned model (Zhang et al., 2021), we

¹We run three times of K-means clustering algorithm and report the average results to avoid the unstable issue in K-means clustering.

have compared two different settings for user utterance representation learning: supervised SimCSE-BERT_base² vs. SimCSE-BERT_large³ (Gao et al., 2021). The results are shown in Table 1, which clearly demonstrate that better user representations lead to more accurate intent discovery. For the compactness of clusters, intuitively, the more compact a cluster is, the more possible user utterances in the cluster share the same intent.

To learn better user utterance representations for clustering and to improve the compactness of learned clusters, we propose CoCoID that learns **C**ontrastive user utterance representations and **C**ompact clusters for semi-supervised **I**ntent **D**iscovery. To learn contrastive representations, we define a user utterance to be clustered as the anchor utterance and feed it to the feature extractor twice with different dropout masks. In doing so, we obtain two different representations for the anchor utterance and use them as positive pairs. Other utterances in the same minibatch serve as negative samples. We then perform contrastive learning to improve utterance representations so that utterances with similar underlying intents tend to be close to each other while utterances with dissimilar intents are separated from each other.

To improve the compactness of clusters, we propose an intra-cluster knowledge distillation (ICKD) method. We randomly sample utterances from the cluster where the anchor utterance locates. The intent label of the anchor utterance is used to guide knowledge distillation from the anchor utterance to the sampled utterances. The motivation behind ICKD is to help shorten the distance between utterances in the cluster, which can be also considered as a regularization strategy for the clustering model.

In summary, our contributions are twofold.

- 1) We propose a novel semi-supervised intent discovery framework CoCoID that learns contrastive user utterance representations and presents an intra-cluster knowledge distillation to improve the compactness of learned intent clusters.
- 2) We have conducted extensive experiments on four challenging datasets, including CLINC, BANKING, StackOverflow and DBPedia, to examine the effectiveness of the proposed Co-

²<https://huggingface.co/princeton-nlp/sup-simcse-bert-base-uncased>

³<https://huggingface.co/princeton-nlp/sup-simcse-bert-large-uncased>

CoID. We have achieved significant improvements over state-of-the-art clustering methods by over 1.4 ACC and ARI and 1.1 NMI points. Further analyses demonstrate that the proposed model can indeed learn contrastive representations and compact clusters.

2 Related Work

Our work is related to intent discovery, contrastive learning and knowledge distillation. We briefly review these topics within the constraint of space.

2.1 Intent Discovery

User intent detection is an essential component in dialogue systems. A wide range of approaches, including unsupervised (Padmasundari and Bangalore, 2018), supervised (Hakkani-Tür et al., 2013) and semi-supervised (Basu et al., 2004; Hakkani-Tür et al., 2015; Han et al., 2019) methods, have been explored for intent discovery. In the unsupervised research strand, Kathuria et al. (2010) propose to exploit K-means clustering to understand user intents. In addition to unsupervised clustering approaches, Haponchyk et al. (2018) solve intent discovery by using powerful semantic classifiers to categorize user questions into intents with structured outputs in a supervised way. Caron et al. (2018) propose to produce clustering assignments as pseudo labels and then train a pseudo classifier. For semi-supervised methods, Zhang et al. (2021) investigate the label inconsistent issue and propose a deep alignment strategy. Other semi-supervised studies approach intent discovery by guiding the clustering process with pairwise constraints, such as KCL (Hsu et al., 2018) and CDAC+ (Lin et al., 2020). Our model is also semi-supervised. The significant differences from previous semi-supervised intent discovery lie in the contrastive learning of user utterance representations and intra-cluster knowledge distillation for improving the compactness of clusters.

2.2 Contrastive Learning

The main idea behind contrastive learning is to force semantic representations of similar objects to be close to each other and those of dissimilar objects to be far away from each other, which is widely used in unsupervised visual representation learning. Recent years have witnessed that contrastive learning has also been explored in textual representation learning. Fang and Xie (2020) pro-

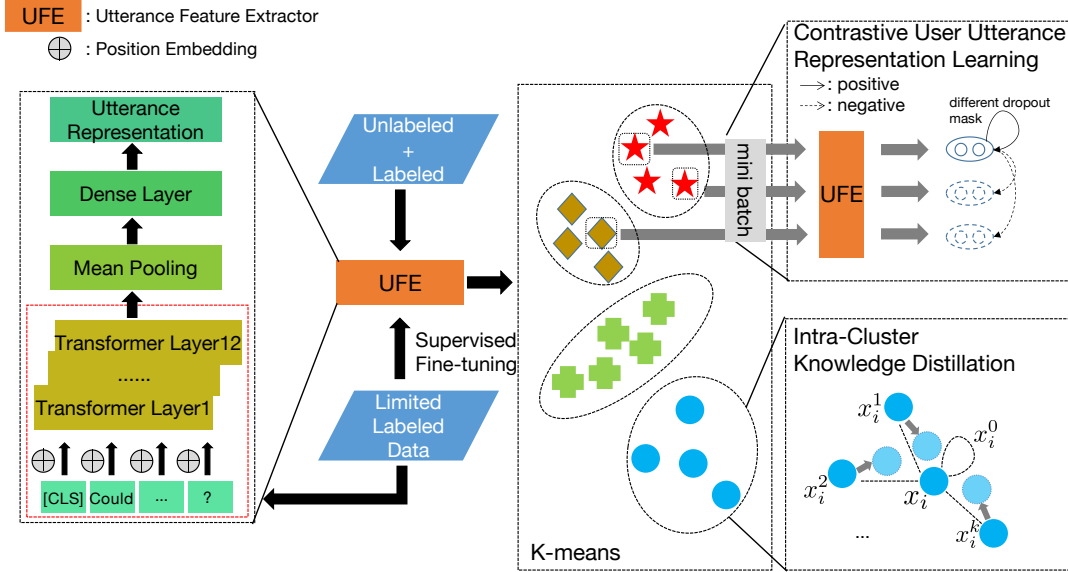


Figure 1: The diagram of CoCoID. We first fine-tune the utterance feature extractor on intent-labeled data. After fine-tuning, we obtain representations for both labeled and unlabeled user utterances. We cluster them via the K-means cluster algorithm. Our contributions lie in the contrastive user utterance representation learning and intra-cluster knowledge distillation used in the semi-supervised clustering procedure.

pose CERT and construct positive and negative utterances through back translation. Gao et al. (2021) find that dropout can act as an efficient data augmentation method for contrastive textual representation learning. Yan et al. (2021) also explore different data augmentation strategies on contrastive learning for sentence representation modeling. In this paper, we follow Gao et al. (2021) to use different dropout masks to create positive samples.

2.3 Knowledge Distillation

Knowledge distillation usually refers to training a large teacher model and distilling its knowledge into a small student model (Hinton et al., 2015). Zhang et al. (2018) propose a strategy where an ensemble of students learn collaboratively and then teach each other. Yuan et al. (2019) find that knowledge distillation is a more general label smoothing regularization and present a teacher-free knowledge distillation framework where a student model learns from itself. Yun et al. (2020) propose a class-wise knowledge distillation method which distills the predictive distribution between different samples of the same label during training, which is similar to our intra-cluster knowledge distillation in the sense of distilling predictive distribution between samples from the same group. The significant difference is that we distill knowledge between user utterances in the same cluster under

the semi-supervised setting rather than in the supervised condition for images (Yun et al., 2020).

3 Approach

We elaborate the proposed CoCoID in this section. The diagram of CoCoID is shown in Figure 1. It consists of three major components: semi-supervised clustering backbone, contrastive utterance representation learning and intra-cluster knowledge distillation.

3.1 Semi-supervised Clustering Backbone

The traditional clustering-based approach for intent discovery produces clustering assignments as pseudo labels (Caron et al., 2018), which may result in an inconsistent clustering assignment issue as different labels could be assigned to the same utterance in different training epochs. We therefore use the DeepAligned framework (Zhang et al., 2021) as our semi-supervised clustering backbone, which is composed of three essential components: utterance feature extractor, fine-tuning and DeepAligned clustering (addressing the inconsistent issue). We briefly introduce the first two parts here. More details on the third component can be found in (Zhang et al., 2021).

Utterance Feature Extractor We use BERT (Devlin et al., 2019) to extract features from user ut-

terances. For notational convenience, we define the training corpus as $\mathcal{D} = \{x_1, x_2, \dots, x_n\}$ where x_i is the i -th utterance with or without intent label. We assume that \mathcal{D} is composed of $\mathcal{D}_{\text{label}}$ and $\mathcal{D}_{\text{unlabel}}$ that indicate utterances with intent labels and those without manual intent labels, respectively. We feed x_i into BERT and extract the last layer of BERT as $\mathbf{H}_i = [\text{CLS}, s_1, s_2, \dots, s_m]$ where CLS is a special classification token and m denotes the length of the utterance x_i . We then obtain an averaged representation for the corresponding utterance as follows:

$$\mathbf{R}_i = f(\mathbf{H}_i) \quad (1)$$

where f indicates the meaning-pooling operation. To further enhance the feature extractor, we add a dense layer g :

$$\mathbf{h}_i = g(\mathbf{R}_i) = \sigma(\mathbf{W}_g \mathbf{R}_i + \mathbf{b}_g) \quad (2)$$

where σ is a ReLU activation function, \mathbf{W}_g and \mathbf{b}_g are learnable parameters.

Fine-tuning To incorporate prior knowledge from limited intent-labeled data, we follow [Zhang et al. \(2021\)](#) to fine-tune the BERT-based feature extractor on labeled data $\mathcal{D}_{\text{label}}$. Specifically, we stack a simple intent classification layer over the BERT-based feature extractor and fine-tune the extractor with a cross-entropy loss in a supervised fashion. After fine-tuning, we remove the classification layer and keep the rest of the network as feature extractor for later use in the clustering process.

3.2 Contrastive User Utterance Representation Learning

After fine-tuning, we obtain a full-fledged feature extractor. We learn representations of utterances in \mathcal{D} through this feature extractor. Over these learned representations, we perform K-means clustering to categorize utterances into groups. We define groups as $\{G\}_{i=1}^K$ where K is the number of groups for clustering.

We define an utterance x_i in question as anchor utterance and its group as $G_{\mu(x_i)}$ where μ is a mapping function that maps x_i to a cluster index $\in [1, K]$. Similar to SimCSE ([Gao et al., 2021](#)), we input each anchor utterance x_i to the encoder (as shown in the red dashed box) twice with two different dropout masks d and d' . We let \mathbf{h}_i^d and $\mathbf{h}_i^{d'}$ (calculated according to Eq. 2) denote the representation of x_i with the two different dropout

masks. We consider $\mathbf{h}_i^{d'}$ as the positive representation to \mathbf{h}_i^d for the anchor utterance since they are only different in dropout masks. That is to say, they are semantically similar to each other. Representations of other utterances in the same minibatch are regarded as negative representations to \mathbf{h}_i^d . In this way, we construct positive and negative representations for contrastive learning. The contrastive learning objective is hence optimized as follows:

$$\mathcal{L}_{\text{CL}} = -\log \frac{\exp(\text{sim}(\mathbf{h}_i^d, \mathbf{h}_i^{d'})/\tau)}{\sum_{j=1}^N (\exp(\text{sim}(\mathbf{h}_i^d, \mathbf{h}_j^{d'})/\tau))} \quad (3)$$

where N is the number of utterances in the minibatch and τ is a temperature hyperparameter. sim is a similarity measurement function, which can be computed as the cosine similarity as follows:

$$\text{sim}(a, b) = \frac{a^\top b}{\|a\| \cdot \|b\|} \quad (4)$$

3.3 Intra-cluster Knowledge Distillation

Dark knowledge can not only be distilled from a large teacher model to a small student model, but also be distillable among continual instances within the same model via self-learning. The latter self-knowledge distillation is feasible as dark knowledge can be regularized to produce the same prediction pattern for instances in the same class ([Yun et al., 2020](#)). This inspires us to perform self-knowledge distillation within the same cluster.

Specifically, for each anchor utterance x_i , we randomly sample different utterances in $G_{\mu(x_i)}$ (denoted as x_i^{in}) to form multiple (x_i, x_i^{in}) pairs. Let \mathbf{u}_i denote logit from x_i and \mathbf{u}_i^{in} logit from x_i^{in} . We then distill dark knowledge from \mathbf{u}_i to each sampled logit. Mean square error (MSE) is used as the intra-cluster distillation objective, which is computed as follows:

$$\mathcal{L}_{\text{ICKD}} = \frac{1}{k+1} \sum_{in=0}^k \|\mathbf{u}_i - \mathbf{u}_i^{in}\|^2 \quad (5)$$

where k is the number of sampled utterances, which is a hyperparameter to be tuned. Note that \mathbf{u}_i^0 is logit from the anchor utterance itself with a different dropout mask.

Forcing the predictive distributions (i.e., logit) from sampled utterances to be close to that from the anchor utterance in the same cluster, we want these utterances in the same cluster to be similar to each other. In other words, their distance could be

Model	CLINC			BANKING			StackOverflow			DBPedia		
	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI	ACC	ARI	NMI
KM†	45.06	26.86	70.89	29.55	12.18	54.57	13.55	1.46	8.24	61.00	49.93	67.26
AG†	44.03	27.70	73.07	31.58	13.31	57.07	14.66	2.12	10.62	56.07	43.92	65.63
SAE-KM†	46.75	29.95	73.13	38.92	22.85	63.79	34.44	17.07	32.62	50.29	31.72	59.70
PCK-means†	54.61	35.40	68.70	32.66	16.24	48.22	24.16	5.35	17.26	83.11	71.27	79.76
DEC†	46.89	27.46	74.83	41.29	27.21	67.78	13.09	3.76	10.88	39.60	29.43	53.36
DCN†	49.29	31.15	75.66	41.99	26.81	67.54	34.26	15.45	31.09	47.48	32.31	54.54
DAC†	55.94	40.49	78.40	27.41	14.24	47.35	16.30	2.76	14.71	63.96	56.30	75.37
BERT-KCL†	68.86	58.79	86.82	60.15	46.72	75.21	13.94	7.81	8.84	60.62	61.03	83.16
DeepCluster†	35.70	19.11	65.58	20.69	8.95	41.77	-	-	-	-	-	-
BERT-DTC†	74.15	65.02	90.54	56.51	44.70	76.55	-	-	-	-	-	-
CDAC+†	69.89	54.33	86.65	53.83	40.97	72.25	73.48	52.59	69.84	91.66	89.41	94.74
DeepAligned†	86.49	79.75	93.89	64.90	53.64	79.56	-	-	-	-	-	-
CDAC+‡	70.18	58.68	87.23	54.34	41.86	72.68	74.8	50.75	76.96	91.31	87.47	93.01
DeepAligned‡	86.69	80.19	94.07	65.44	53.76	79.80	78.55	61.21	75.97	92.89	89.54	94.04
CoCoID (ours)	87.51	81.35	94.49	67.81	57.08	81.32	78.43	60.25	77.60	95.56	91.91	94.95
w/o ICKD	86.74	80.31	94.06	66.44	54.83	80.26	79.10	60.13	77.46	95.45	91.52	94.72
w/o CL	86.13	80.7	94.53	66.52	56.51	81.75	78.35	59.04	77.52	93.74	90.34	94.37

Table 2: Clustering results on the four datasets. †: results from Zhang et al. (2021) and Lin et al. (2020). ‡: results that we reproduced.

shortened by the proposed intra-cluster knowledge distillation so that clusters are more compact.

4 Training Objective

Contrastive utterance representation learning attempts to pull utterances with similar semantic intents together and push apart utterances with different intents in a minibatch-wise manner. By contrast, intra-cluster knowledge distillation is to shorten distances of utterances in the same cluster by regularizing predictive distributions in a cluster-wise way. These two approaches are complementary to each other and therefore can be combined in a unified framework CoCoID. The final joint loss of CoCoID with the two components can be formulated as:

$$\mathcal{J} = \mathcal{L}_{CE} + \mathcal{L}_{CL} + \mathcal{L}_{ICKD} \quad (6)$$

where \mathcal{L}_{CE} is the cross-entropy loss used in the DeepAligned model (Zhang et al., 2021).

5 Experiments

We conducted a series of experiments on four widely-used datasets to evaluate the proposed CoCoID. More details about the datasets and baselines can be found in the appendix A and B.

5.1 Evaluation Metrics

We used three metrics to evaluate performance in our experiments. Normalized Mutual Information (NMI) is commonly used in measuring the quality

of clustering by estimating the similarity of clustering results to the ground-truth results. Adjusted Rand Index (ARI) treats the analysis of clustering as a series of decisions, one for each of the $N(N-1)/2$ pairs of collections. The Rand index measures the percentage of decisions that are correct while ARI is the corrected-for-chance version of it, ensuring that the value for random clustering tends to be 0. In addition to the two metrics, we follow Zhang et al. (2021) to use the Hungarian algorithm to obtain the mapping between the predicted classes and ground-truth classes to estimate clustering Accuracy (ACC). For all metrics, the higher the score is, the better the performance is.

5.2 Settings

To make fair comparisons, we used the same settings that randomly select 10% of the data as the labeled data, and 75% of the intents as the known intents as in previous works (Zhang et al., 2021). The division of the datasets also follows Zhang et al. (2021) and Lin et al. (2020). We first fine-tuned our model on the labeled data and performed contrastive learning and intra-cluster knowledge distillation during clustering. We set the wait patient as 20 to avoid overfitting. We evaluated the cluster performance with Silhouette Coefficient which is an unsupervised metric to evaluate clustering performance. After training, we evaluated the performance on test sets and averaged 10 random-seed results as our final result. Note that we set the num-

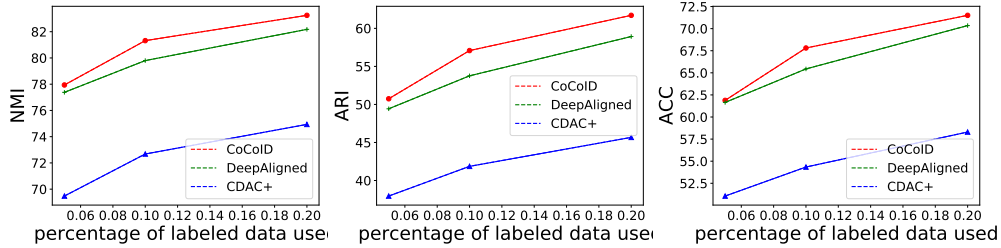


Figure 2: Comparison results along varying labeled data ratios on the BANKING dataset.

ber of intents as ground-truth during clustering, which is consistent with Zhang et al. (2021).

We employed the pre-trained BERT model (base-uncased, with 12-layer Transformer) to build the feature extractor. We set the training batch size as 128, and the learning rate as $5e^{-5}$. The temperature for contrastive utterance representation learning was set to 1.0 for CLINC, StackOverflow and DBPedia dataset. For BANKING dataset, we set the temperature of contrastive learning as 0.05. We also followed Zhang et al. (2021) to freeze all BERT parameters except the last layer to avoid overfitting and speed up the training process. For each anchor utterance, we sampled 3 utterances.

5.3 Main Results

Table 2 shows the results on the four datasets. Our reproduced CDAC+ and DeepAligned model are better than those reported by Zhang et al. (2021) and Lin et al. (2020) in most cases, indicating strong baselines to be compared. Our proposed CoCoID outperforms the 12 state-of-the-art baselines on the majority of cases on the four datasets. Particularly, we achieve an average improvement of 1.43 ACC, 1.47 ARI and 1.12 NMI over the reproduced DeepAligned (better than the original).

Interestingly, it is worth noting that the length of utterances in datasets seems to have an impact on the performance improvements. The average length of utterances in both CLINC and StackOverflow datasets is relatively shorter than 10, hence the improvement obtained by our proposed model is limited. In contrast, on the BANKING and DBPedia datasets, utterances are longer. The improvements in terms of the three metrics are substantially higher than those on the other two datasets.

5.4 Ablation Study

To examine the effectiveness of the proposed two methods, we further conducted ablation study. In the last two rows of Table 2, we show the results of

our model without contrastive learning and intra-cluster distillation. It can be found that the removal of contrastive learning will reduce the ACC of the model to some extent, while the removal of ICKD will negatively affect the NMI and ARI metric. The absence of contrastive learning causes ACC to decrease by 1.14 on average, suggesting that contrastive learning is able to boost intent discovery accuracy. The absence of intra-cluster knowledge distillation results in larger performance drops in CLINC and BANKING than those in StackOverflow and DBPedia. This indicates that ICKD is beneficial to intent discovery with a larger number of novel intents (37/19 vs. 5/4)

6 Analysis

We carried out in-depth analyses to investigate how the proposed methods improve intent discovery.

6.1 Analysis on the Impact of the Percentage of Labeled Data Used

As our CoCoID is a semi-supervised intent discovery approach that utilizes intent-labeled data to fine-tune the feature extractor, we would like to know the impact of the percentages that used intent-labeled data account for among all data on the performance. For this, we conducted experiments on the BANKING dataset by gradually increasing the percentage of labeled data used for fine-tuning from 0.05 to 0.2. Results of CoCoID against the two state-of-the-art models DeepAligned and CDAC+ are illustrated in Figure 2. Under all settings, all the three models benefit from the growing amount of intent-labeled data used. But our CoCoID is always the best among the three models, indicating its strong capacity in exploring intent-labeled data.

6.2 Analysis on the Impact of the Ratio of Unknown Intent Classes

Our approach is able to detect both known (pre-defined) and unknown (novel) intent classes. There-

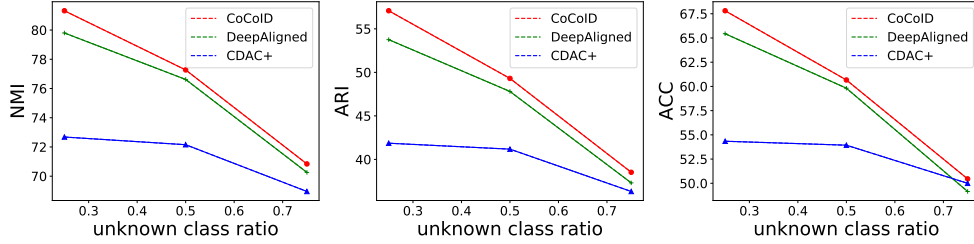


Figure 3: Comparison results along varying known class ratios on the BANKING dataset.

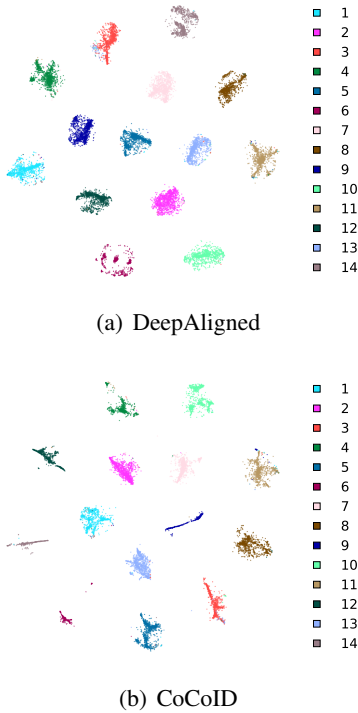


Figure 4: Visualization of the 14 intents on the DBpedia dataset and 5, 6, 9, 13 are the unknown intents.

fore, we further analyzed the impact of the unknown intent class ratio on the performance. Figure 3 shows the results of CoCoID in comparison to CDAC+ and DeepAligned. The performance of the three models drops when the ratio of unknown intent classes increases, which is reasonable as it is more challenging to detect novel intents without labeled data than predefined intents with labeled data. However, our model achieves smaller performance drops in all three evaluation metrics along growing unknown intent class ratios, suggesting that our model is more capable of detecting novel intents than both DeepAligned and CDAC+.

6.3 Visualization of Intent Distribution

Figure 4 visualizes the distribution of 14 intents in the semantic space of the DBpedia dataset. We

merged the training data and test data to generate utterance representations. We then visualized them through t-sne. To obtain a better global structure, we set the perplexity of t-sne to 500. Figure 4(a) visualizes the intent distribution yielded by DeepAligned while figure 4(b) displays results yielded by our CoCoID. Note that the same color across the two sub-figures represents the same intent cluster. From the visualization, we can easily find that the area of intent clusters produced by CoCoID is smaller than that by DeepAligned. And some clusters are even compacted into strips (e.g., intent 3 and 9 in Figure 4(b)).

In order to explicitly show the degree of compactness of intent clusters, we also calculated the average distance from each utterance to the cluster centroid. The average distance for DeepAligned is 0.34 while only 0.14 for our CoCoID, which strongly suggests that our proposed method does make clusters more compact.

7 Conclusions

In this paper, we have presented a semi-supervised intent discovery framework CoCoID. It consists of two essential components: contrastive user utterance representation learning and intra-cluster knowledge distillation. Extensive experiments and analyses on real-life challenging datasets demonstrate that CoCoID outperforms previous state-of-the-art intent discovery models (12 baselines) and is able to learn contrastive representations and compact clusters.

Acknowledgments

Deyi Xiong was supported by Zhejiang Lab (No. 2022KH0AB01). We would like to thank the anonymous reviewers for their insightful comments.

References

- Sugato Basu, Arindam Banerjee, and Raymond J Mooney. 2004. Active semi-supervision for pairwise constrained clustering. In *Proceedings of the Fourth SIAM International Conference on Data Mining*, pages 333–344.
- Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149.
- Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. *CoRR*, abs/2003.04807.
- Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2017. Deep adaptive image clustering. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5880–5888. IEEE.
- Jackie Chi Kit Cheung and Xiao Li. 2012. Sequence clustering and labeling for unsupervised query intent discovery. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 383–392.
- 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.
- Hongchao Fang and Pengtao Xie. 2020. CERT: contrastive self-supervised learning for language understanding. *CoRR*, abs/2005.12766.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910.
- Dilek Hakkani-Tür, Asli Celikyilmaz, Larry P. Heck, and Gökhan Tür. 2013. A weakly-supervised approach for discovering new user intents from search query logs. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 3780–3784. ISCA.
- Dilek Hakkani-Tür, Yun-Cheng Ju, Geoffrey Zweig, and Gökhan Tür. 2015. Clustering novel intents in a conversational interaction system with semantic parsing. In *INTERSPEECH 2015, 16th Annual Conference of the International Speech Communication Association, Dresden, Germany, September 6-10, 2015*, pages 1854–1858. ISCA.
- Kai Han, Andrea Vedaldi, and Andrew Zisserman. 2019. Learning to discover novel visual categories via deep transfer clustering. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8400–8408. IEEE Computer Society.
- Iryna Haponchyk, Antonio Uva, Seunghak Yu, Olga Uryupina, and Alessandro Moschitti. 2018. Supervised clustering of questions into intents for dialog system applications. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2310–2321.
- Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.
- Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. 2018. Learning to cluster in order to transfer across domains and tasks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.
- Ashish Kathuria, Bernard J. Jansen, Carolyn Theresa Hafernik, and Amanda Spink. 2010. Classifying the user intent of web queries using k-means clustering. *Internet Res.*, 20(5):563–581.
- Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. 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 1311–1316.
- Ting-En Lin, Hua Xu, and Hanlei Zhang. 2020. Discovering new intents via constrained deep adaptive clustering with cluster refinement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8360–8367.
- Padmasundari and Srinivas Bangalore. 2018. Intent discovery through unsupervised semantic text clustering. In *Proceedings of Interspeech*, pages 606–610.
- Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, pages 478–487.
- Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 62–69.
- Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of the 59th Annual*

Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 5065–5075.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards k-means-friendly spaces: simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3861–3870.

Li Yuan, Francis E. H. Tay, Guilin Li, Tao Wang, and Jiashi Feng. 2019. Revisit knowledge distillation: a teacher-free framework. *CoRR*, abs/1909.11723.

Sukmin Yun, Jongjin Park, Kimin Lee, and Jinwoo Shin. 2020. Regularizing class-wise predictions via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13876–13885.

Hanlei Zhang, Hua Xu, Ting-En Lin, and Rui Lyu. 2021. Discovering new intents with deep aligned clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14365–14373.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. 2018. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328.

Dataset	#Intents (Known+Unknown)	#Training	#Validation	#Test	Vocabulary	Length (max/mean)
CLINC	150 (113 + 37)	18,000	2,250	2,250	7,283	28/8.31
BANKING	77 (58 + 19)	9,003	1,000	3,080	5,028	79/11.91
StackOverflow	20 (15 + 5)	18,000	1,000	1,000	17,182	41/9.18
DBPedia	14 (10 + 4)	12,600	700	700	45,077	54/29.97

Table 3: Statistics on the used datasets. Intents are divided into known (predefined) and unknown (novel intents).

A Datasets

Detailed statistics of the four datasets are shown in Table 3.

CLINC is a multi-domain intent classification dataset (Larson et al., 2019), which contains 150 intents and 23,700 utterances across 10 domains. To compare with previous works, we used the same data division as in (Zhang et al., 2021).

BANKING is a single-domain intent classification dataset (Casanueva et al., 2020), which provides a fine-grained set of intents in the banking domain. It contains 13,083 utterances labeled with 77 intents. In our experiments, the split of training, validation and test set follows Zhang et al. (2021).

StackOverflow is an intent classification dataset collected and processed by Xu et al. (2015) from Kaggle.com.⁴ Xu et al. (2015) randomly select 20,000 question titles from 20 different labels. In our experiments, the division of training, validation and test set follows Lin et al. (2020).

DBPedia is a DBpedia ontology dataset which contains 14 non-overlapping classes (Zhang and Le-Cun, 2015). We follow Lin et al. (2020) on data division to compare with their results fairly.

B Baselines

We compared with 12 different baselines, listed as follows.

K-means (KM): an iterative algorithm of clustering, which first randomly selects K objects as the initial cluster centroids and then assigns each object to the nearest cluster.

Agglomerative Clustering (AG): a bottom-up hierarchical clustering method which calculates the distance/similarity between classes.

SAE-KM: a method similar to K-means. The difference is that the feature extractor is a stacked autoencoder (SAE).

⁴<https://www.kaggle.com/c/predict-closed-questions-onstack-overflow/download/train.zip>.

PCK-means (Basu et al., 2004): a clustering framework with paired constraints and a new theoretically motivated method that actively selects good paired constraints for semi-supervised clustering.

DEC (Xie et al., 2016): a method that uses self-training target distribution to iteratively optimize clustering targets according to KL divergence.

DCN (Yang et al., 2017): a method that combines a dimension reduction and K-means clustering approach to maintain the advantages of both tasks.

DAC (Chang et al., 2017): a method that converts the clustering problem to a binary pair classification to determine whether a pair of images belongs to the same cluster.

BERT-KCL (Hsu et al., 2018): an approach using predictive pairwise similarity as the knowledge to be transferred, and formulating a learnable objective function to utilize pairwise information in a manner similar to constrained clustering.

DeepCluster (Caron et al., 2018): a clustering method which jointly learns the parameters of the neural network and the cluster assignments from extracted features.

BERT-DTC (Han et al., 2019): a modified variant of DEC (Xie et al., 2016) which can cluster data during learning data representations. The purpose of this change is to allow clustering to be guided by known classes.

CDAC+ (Lin et al., 2020): an end-to-end clustering method that can naturally use paired constraints as prior knowledge to guide the clustering process.

DeepAligned (Zhang et al., 2021): a semi-supervised intent discovery method based on DeepCluster, which uses an alignment strategy to tackle the label inconsistency issue.

Limitations

For the combination of multi-task losses, we interpolate them with equal weights. We believe that there are better strategies to combine these losses

which we leave to our future work. Additionally, the impact of query length on the performance is yet to be investigated.

Tractable & Coherent Multi-Document Summarization: Discrete Optimization of Multiple Neural Modeling Streams via Integer Linear Programming

Litton J Kurisinkel, Nancy F. Chen

Institute for Infocomm Research, A*STAR, Singapore
litton_kurisinkel, nfychen@i2r.a-star.edu.sg

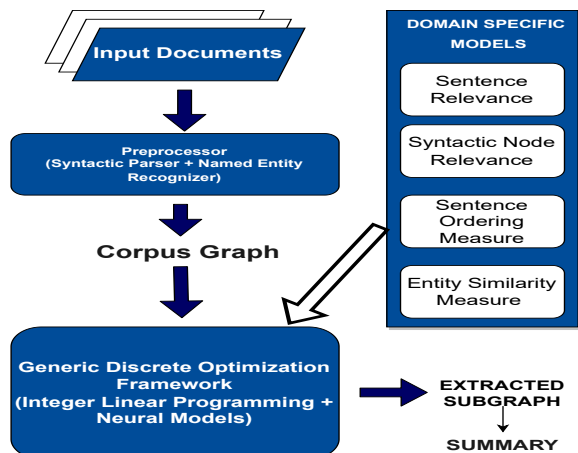


Figure 1: System Architecture(Ref. Figure 2 for a more detailed view)

Abstract

One key challenge in multi-document summarization is the generated summary is often less coherent compared to single document summarization due to the larger heterogeneity of the input source content. In this work, we propose a generic framework to jointly consider coherence and informativeness in multi-document summarization and offers provisions to replace individual components based on the domain of source text. In particular, the framework characterizes coherence through verb transitions and entity mentions and takes advantage of syntactic parse trees and neural modeling for intra-sentential noise pruning. The framework cast the entire problem as an integer linear programming optimization problem with neural and non-neural models as linear components. We evaluate our method in the news and legal domains. The proposed approach consistently performs better than competitive baselines for both objective metrics and human evaluation.

1 Introduction

Multi-Document summarization (MDS) approaches generate the summary of a corpus of

documents consisting of a set of related topics. Extractive summarization techniques extract a subset of sentences which topically represent the input corpus in a stipulated summary space (Lin and Bilmes, 2011). On the other hand, abstractive summarization techniques constructs a semantic representation of the source text and constructs the summary in its own learnt writing style (Tan et al., 2017). Despite the attempts for abstractive summarization techniques using neural methods, extractive summarization techniques reserve its space for formulating ready to use summarization approaches. However, a set of (selected) sentences put together without considering the ordering and coherence of the content may not make much sense to the summary reader (Guinaudeau and Strube, 2013; Barzilay and Lapata, 2008). Hence, the to make such text generation applications more accessible to users, it is essential to improve qualitative dimensions such as *coherence*. Studies on human written summaries shows that generic information is more relevant for summary content while more specific information is considered to be irrelevant (Louis and Nenkova, 2011). In MDS, the unit of extraction is sentences. Long sentences could often contain irrelevant information that is not essential to the summary and thus regarded as *noisy* information (Knight and Marcu, 2000). Hence MDS systems should ideally be equipped for pruning intra-sentential noise.

Most previous work for extractive MDS gave less importance in improving summaries in qualitative dimensions such as coherence and provided an incoherent reading to the summary reader (Takamura and Okumura, 2009). A subset of previous work aimed at removing intra-sentential noise by sentence compression (Berg-Kirkpatrick et al., 2011). Such works were successful in removing intra-sentential noise, however the problem of incoherent reading can be more severe as the parts of the sentences pruned away can be important

Input Document Sentences

S₁) Alberta Press Act Reference(1938), also called Reference Re Alberta Legislation, concerned 4 Alberta statutes, one of which, the Accurate News and Information Act, would have compelled each newspaper in the province, when called upon to do so to a government official, to publish the government's rebuttal of criticism that had appeared in the newspaper.
 S₂) The new bills in question were the Bank Taxation Act, an Act to Amend the Credit of Alberta Regulation Act and the Act to Ensure the Publication of Accurate News and information.
 S₃) On October 6, 1937, Lieutenant Governor Bowen announced that he was reserving Royal Assent on the three bills until they could be sent to the Supreme Court for review.

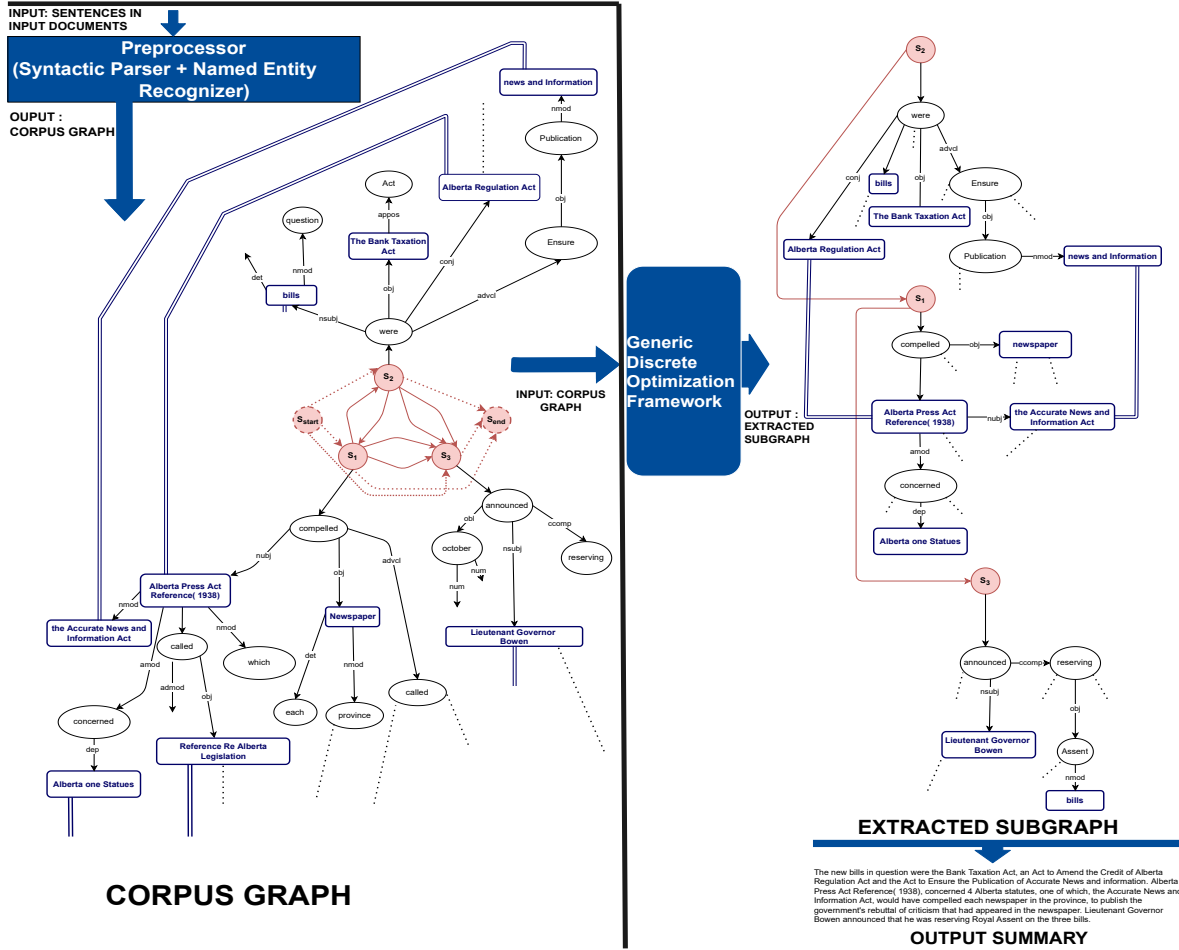


Figure 2: Flowchart for Coherent and Noise-Free Multi-Document Summarization with Input and Intermediate and Final Outputs: Figure depicts the summarization of an input corpus containing three sentences. Left side of the figure depicts the corpus graph constructed while right portion depicts the extracted sub-graph. Sentence nodes and sentence pair edges are depicted in red, syntactic tree nodes and edges are depicted in black and entity nodes, and entity pair edges are depicted in blue.

for topical continuity and for coherence. Very few works formulated methods to improve topical coherence between sentences in the summary by constructing corpus level discourse graph or by computing entity transition probabilities (Christensen et al., 2013; Wang et al., 2016). However such works had no mechanism for pruning intra-sentential noise. Through the current work we present a hybrid approach using Integer Linear Programming (ILP) and neural models which jointly does relevant and informative content selection, sentence compression for *noise pruning* and content ordering for *coherence*. We investigate for a framework as shown in Figure 1 which gives good cross-domain performance with minimal replacement of components.

2 Related Work

Text summarization can be achieved using extractive (Takamura and Okumura, 2009; Lin and Bilmes, 2011; Wang et al., 2008) and abstractive methods (Bing et al., 2015; Li, 2015). Extractive summarization has the advantage of output fluency due to direct use of human-written texts. However, because of the higher level of granularity exhibited by sentences, these approaches cannot ensure a noise free and coherent summary. A subset of previous extractive summarization approaches utilized parsed sentence structures to execute noise pruning while extracting content for summary (Morita et al., 2013; Berg-Kirkpatrick et al., 2011). But these techniques can merely prune noise, and can-

not ensure a coherent reading for the summary reader. The attempt to achieve *coherence* in multi-document summarization was attempted by some of the extractive summarization system. (Christensen et al., 2013; Wang et al., 2016). However the attempts to achieve coherence in an MDS scenario often cope with intra- sentencial noise for coherence.

3 Method

Approach: We formulate an approach which ensures the construction of multi-document extractive summaries encompassing relevant and coherent content as depicted using Figures 1 and 2. The approach involves two steps, represented by blue rectangular boxes in the figure.

- **Preprocessing:** Identify named entities in the documents and parse the sentences in the input set of documents using a syntactic dependency parser. Construct a ‘Corpus Graph’ which provides means for tracking coherent and relevant content and for noise removal.
- **Summary Extraction:** Extract a sequence of noise- pruned sequence of syntactic subtrees which hold coherent, relevant and grammatically accurate information. The sequence of subtrees can be directly linearized to a coherent sequence of sentences.

The following subsections explain each one of these steps in detail.

3.1 Preprocessing and Corpus Graph Construction

At this stage, named entities in the input corpus sentences are identified and chunked ¹. Subsequently, the sentences in the input corpus is parsed into syntactic dependency trees ². The set of syntactic dependency trees are transformed into a corpus graph by adding extra nodes and edges (Figure 2). The graphs contains Sentence nodes (S_i) and Dummy start (S_{start}) and end nodes (S_{end}), Sentence Pair Edges (E_{ij}), Syntactic Trees Nodes N_{ij} , Named Entity Node N_{ij} , Entity Pair Edges ($EE_{ij,mn}$) represented by red colored circles, red colored arrows, ellipses drawn in black lines, rectangles drawn in blue colored lines and blue colored double lines respectively.

¹<https://stanfordnlp.github.io/CoreNLP/ner.html>

²<https://nlp.stanford.edu/software/lex-parser.shtml>

3.2 Summary Extraction

Using Corpus Graph, we tranform summary extraction into a sub- graph extraction problem. The extracted subgraph SG should be of the form of the graph depicted at the right portion of the Figure 2. The extract should represent a sequence of sentence nodes with syntactic subtrees containing the most salient information attached to them. The syntactic subtrees are formed by removing noisy portions of the original syntactic trees. The sequence should maximize coherence quantified by the selected set of sentence pair and entity pair edges. The total size of the text content held by the selected sequence of subtrees should be within the specified summary size. We formulate the sub- graph extraction from corpus graph as an integer linear programming(ILP) problem. Our ILP formulation is given below.

Maximize,

$$\begin{aligned}
F(X) = & \lambda_1 \sum_{S_i \in SN} SSal(S_i) * sx_i + \\
& \lambda_2 \sum_{n_{ij} \in TN} NSal(n_{ij}) * nx_{ij} + \\
& \lambda_3 \sum_{e_{ij,ik} \in TE} ESal(e_{ij,ik}) * ey_{ij,ik} + \\
& \lambda_4 \sum_{e_{ij,ik} \in EE} eSim(ne_{ij}, ne_{ik}) * sy_{ij,ik} + \\
& \lambda_5 \sum_{E_{ij} \in SE} Prob(S_i, S_j) * Ey_{ij} - \\
& \lambda_6 \sum_{E_{ij} \in SE} SSim(S_i, S_j) * Ey_{ij}
\end{aligned} \tag{1}$$

Subject to Constraints,

$$\begin{aligned}
& \forall S_i \in S, S_j \in S \\
& 2 * (Ey_{ij} + Ey_{ji}) - (Sx_i + Sx_j) \leq 1(c_1)
\end{aligned}$$

$$\forall S_i, \sum_j Ey_{ij} = 1, \sum_j Ey_{ji} = 1(c_2)$$

$$\sum_{S_i \in S} Ey_{start,i} = 1, \sum_{S_i \in S} Ey_{i,end} = 1(c_3)$$

$$\sum_j Ey_{ij} - \sum_j Ey_{ji} = 0(c_4)$$

$$\sum_{S_i \in S} Sx_i + \sum_{E_{ij} \in SE} Ey_{ij} = 0(c_5)$$

$$\forall N_{ij} \in TN, Sx_i - nx_{ij} \geq 0(c_6)$$

$$\forall S_j \in S, \sum_{N_{ij} \in TN} nx_{ij} - Sx_i \geq 0(c_7)$$

$$\forall e_{ij,ik} \in TE$$

$$nx_{ij} + nx_{ik} - 2ey_{ij,ik} \leq 1(c_8)$$

$$nx_{ij} + nx_{ik} - 2ey_{ij,ik} \geq 0(c_9)$$

if $\text{DepReIn}(e_{ij,ik}) \in \text{GramRelns}$

$$nx_{ij} - nx_{ik} = 0(c_{10})$$

$$\forall S_i \in S,$$

$$sx_i - nx_{iroot} = 0(c_{11})$$

$$sx_i - \left(\sum_{N_{ij} \in TN} nx_{ij} - \sum_{N_{ij} \in TE} ey_{ij,ik} \right) = 0(c_{12})$$

$$\forall n_{ij} \in TN, \sum_{e_{ij,ik} \in TE} ey_{ij,ik} - nx_{ij} (c_{13})$$

$$\forall e_{ij,mn} \in EE,$$

$$2 * sy_{ij,mn} - (nx_{ij} + nx_{mn}) \geq 0(c_{14})$$

$$2 * sy_{ij,mn} - (nx_{ij} + nx_{mn}) \leq 1(c_{15})$$

$$(Ey_{im} + Ey_{mi}) - sy_{ij,mn} \geq 0(c_{16})$$

$$\sum_{n_{ij} \in TN} size(n_{ij}) * nx_{ij} \leq SumSize(c_{17})$$

$$\sum_{S_i \in S} sx_i \leq N(c_{18})$$

Where,

$\mathbf{X} = (.sx_i., .Ey_{ij}., .nx_{ij}., .ey_{ij,ik}., .sy_{ij,ik}.)$ represents a binary indicator vector corresponding to a candidate subgraph SG to be extracted. SN is the set of sentence nodes, SE is the set of sentence pair edges, TN is the set of syntactic tree nodes, TE is the sentence tree edges and EE is the set of entity pair edges. Indicator variables in SG represents different components of the graph as

follows.

3.2.1 Linear Components of F:

$SSal$, $NSal$ and $ESal$ computes salience of sentence, node and edge respectively. $SSim$ computes the similarity between sentences while $ESim$ computes similarity between entities. $Prob$ returns the transition probability of main verbs of parameter sentences, pre-computed using a large domain specific corpus. $SSim$ is used to penalize the redundant content in candidate summaries. $ESim$ and $Prob$ contributes for encouraging coherence of the summary to be extracted.

3.2.2 ILP Constraints:

The constraints c_1 to c_5 ensures the consistency between sentences nodes and sentence pair edges in SG . Also ensures that extracted subgraph contains a sequence of sentence nodes. c_6 to c_9 ensures the consistency of selection between sentence nodes, tree nodes and tree edges. $DepReIn$ return the dependency relation corresponding to the parameter tree edge and $GramRelns$ contains the dependency relations required ensure grammaticality. The constraint c_{10} ensures that nodes which are essential for a syntactic subtree to hold grammatically accurate information won't be pruned away. n_{iroot} indicates the root node of the syntactic tree corresponding to S_i and the constraint c_{11} ensures that subtrees extracted are rooted at the original root node. Constraints c_{12} to c_{16} ensures that entity pair edges are active only when corresponding named entity nodes are selected and when corresponding sentences are neighbours in the sequence of sentence nodes contained in SG .

4 Experiments and Results

4.1 Data

We evaluate our method using the test sets of DUC 2004³ and corpus MDS testset released by (Zopf et al., 2016) for law & Politics domain. We resort to standard ROUGE metric (Lin, 2004) for measuring content selection and rely on human evaluation for measuring coherence and linguistic quality. We tune our hyper parameters using the DUC 2003 dataset⁴.

³<http://duc.nist.gov/data.html>

⁴<http://duc.nist.gov/data.html>

System	D-2004(News)				Law			
	R-1	R-2	R-L	R-W	R-1	R-2	R-L	R-W
Lin and Bilmes (2011)	39.3	10.7	38.7	15.7	41.4	9.7	40.75	21.90
Berg-Kirkpatrick et al. (2011)	36.3	8.3	37.3	13.7	40.14	9.12	39.91	21.70
Bing et al. (2015)	34.0	7.3	33.0	11.6	41.3	8.09	41.3	21.12
Christensen et al. (2013)	37.3	8.2	37.0	13.9	36.25	7.18	37.0	18.16
Wang et al. (2016)	39.0	9.3	37.3	13.7	39.30	8.75	38.25	19.30
Current System + G-Flow	38.3	9.8	38.0	13.6	40.7	9.33	40.91	21.37
Current System + BertSum	37.7	9.3	37.7	12.9	-	-	-	-

Table 1: Comparison with state of the art. In the Table R represents Rouge

	Coh			Inf	Gram				
	PS	OS	AMB		PS	OS	AMB		
Peer Systems(PS)									
Lin and Bilmes (2011)	21	79	0	60	30	10	63	29	8
Berg-Kirkpatrick et al. (2011)	19	72	9	32	45	23	31	37	32
Bing et al. (2015)	20	67	13	34	60	6	37	40	23
Christensen et al. (2013)	43	54	3	18	60	12	70	17	13
Wang et al. (2016)	40	52	8	30	61	9	70	27	3
Kappa	73			77			72		

Table 2: Human Evaluation: In the Table, PS is Peer System, OS is Our System, Amb is Ambiguous, Coh is Coherence, Inf is Informativeness and Gram is Grammaticality

4.2 Settings

4.2.1 Saliency Functions

- *SSal*: To compute sentence saliency we use the same linear regression function proposed by Christensen et al. (2013). For news domain, we also leverage BertSum (Liu and Lapata, 2019) for computing sentence relevance.
- *NSal*: To compute syntactic tree node saliency we use the neural model proposed by Kurisinkel et al. (2019) which leverage syntactic context information to compute the saliency of a node.
- *ESal*: We set the weight of syntactic tree edge e as the frequency of D_{bigram} which is the bigram constituted by the words incident on e .

4.2.2 Similarity Functions: *SSim* & *ESim*

We rely on overlapping words for entity similarity and the similarity is computed using Jaccards Index (Hamers et al., 1989) entity word sets. For sentence similarity we rely on the method suggested by (Pawar and Mago, 2018). They compute the semantic similarity between sentences based on word similarity, sentence similarity and word order similarity.

4.2.3 Verb Transition Probability: *Prob*

We learn the fully connected neural network to learn verb transition probabilities. To learn the probabilities, we collect corpus of 16000 and 4500

documents in news and legal domains respectively. We extract main verbs from each sentence in the document using Stanford Parser⁵.

5 Evaluation

5.1 Evaluation of Content Coverage

We evaluated content coverage of the summary using objective metrics such as ROUGE. As show in the Table 1, results are reported in terms of ROUGE-1, ROUGE-2 and ROUGE -L. (Lin and Bilmes, 2011) consistently performed well in terms of ROUGE score. They incorporate a monotone sub-modular scoring function which is designed for quantitatively improving content coverage. Sub-modular maximization functions cannot be incorporated in an ILP setting with provision for improving coherence. Our approach yielded results that is comparable with (Lin and Bilmes, 2011) while out performing most of the other systems considered for evaluation. Other systems which incorporated coherence (Wang et al., 2016; Christensen et al., 2013) did not perform well in the evaluation for content coverage. We observe that these systems compromised on relevant content without any means for removing intra- sentential noise. However, our approach for coherent summarization incorporates means for intra- sentential noise pruning performed well in terms of ROUGE evaluation.

⁵<https://nlp.stanford.edu/software/lex-parser.shtml>

System Summary
The new bills in question were the Bank Taxation Act, and the Act to Ensure the Publication of Accurate News and information. Alberta Press Act Reference-LRB- 1938-RRB-, concerned 4 Alberta statutes, one of which, the Accurate News and Information Act, would have compelled each newspaper in the province, to publish the government’s rebuttal of criticism that had appeared in the newspaper. Coverage by the Edmonton Journal earned the newspaper a special Pulitzer Prize“ for its editorial leadership in defense of the freedom of the press.
Original Sentences
The new bills in question were the Bank Taxation Act, an Act to Amend the Credit of Alberta Regulation Act and the Act to Ensure the Publication of Accurate News and information. Alberta Press Act Reference-LRB- 1938-RRB-, also called Reference Re Alberta Legislation, concerned 4 Alberta statutes, one of which, the Accurate News and Information Act, would have compelled each newspaper in the province, when called upon to do so by a government official, to publish the government’s rebuttal of criticism that had appeared in the newspaper. Coverage by the Edmonton Journal was particularly strong and eventually earned the newspaper a special Pulitzer Prize“ for its editorial leadership in defense of the freedom of the press.
Reference Summary
The Accurate News and Information Act was a statute passed by the Legislative Assembly of Alberta, Canada, in 1937, at the instigation of William Aberhart’s Social Credit government. It would have required newspapers to print "clarifications" of stories that a committee of Social Credit legislators deemed inaccurate, and to reveal their sources on demand. The act was a result of the stormy relationship between Aberhart and the press, which dated to before the 1935 election, in which the Social Credit League was elected to government.

Table 3: Tree Combination vs Phrase Combination

5.2 Human Evaluation

We conducted human evaluation for evaluating summaries in other qualitative dimensions such as coherence, grammaticality and informativeness. Evaluators are four post graduate students in linguistics. During each evaluation process one among the peer systems compete with our system. 20 summaries generated by each one of the systems competing systems are chosen for evaluation. Summaries are shown to the evaluators in random order to avoid any kind of bias. For evaluating coherence and grammaticality, for each summary pair competing summaries, evaluators are asked to choose the best one in terms of the aspect under evaluation. For informativeness, they are asked to read the reference summary and asked to choose the most informative one. Results are shown in the Table 2. Our approach performed consistently better than other systems in the evaluation for coherence. We used verb transition probability in combination with entity similarity for modelling coherence. We observe this as the reason for our better performance in comparison with (Christensen et al., 2013) and (Wang et al., 2016). Our method performed comparably with other systems in the evaluation for informativeness. Obviously

the methods which don’t modify the original sentences performed better than our method in the evaluation for grammaticality. However we performed better than (Berg-Kirkpatrick et al., 2011). This shows that explicit use of neural model for computing node relevance using syntactic context and grammatical rules based on syntactic relations helped in maintaining grammaticality.

6 Discussions

A system summary generated for an input corpus in Law domain is shown in the Table in the next page. The table also contains the sequence of original sentences in the corpus with intra- sentential noisy information and the corresponding reference summary. Clearly our method were successful in removing intra- sentential noise and organizing summary for a coherent ordering. The neighboring sentences contained similar entities and the order of main verbs (*were, compelled, was*) is the most likely one as per the transition probabilities computed using neural model for verb transition. The summary is also infomative as per the human written abstractive reference summary

Acknowledgements

This research was partially supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. We would also like to thank anonymous reviewers for several very insightful feedback on how to improve the paper.

References

- Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.
- Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.
- Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*.
- Janara Christensen, Stephen Soderland, Oren Etzioni, et al. 2013. Towards coherent multi-document summarization. In *Proceedings of the 2013 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 1163–1173.
- Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 93–103.
- Lieve Hamers et al. 1989. Similarity measures in scientific research: The jaccard index versus salton's cosine formula. *Information Processing and Management*, 25(3):315–18.
- Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI*, 2000:703–710.
- Litton J Kurisinkel, Yue Zhang, and Vasudeva Varma. 2019. Domain adaptive neural sentence compression by tree cutting. In *European Conference on Information Retrieval*, pages 475–488. Springer.
- Wei Li. 2015. Abstractive multi-document summarization with semantic information extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1908–1913.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 510–520. Association for Computational Linguistics.
- Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.
- Annie Louis and Ani Nenkova. 2011. Text specificity and impact on quality of news summaries. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 34–42.
- Hajime Morita, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2013. Subtree extractive summarization via submodular maximization. In *ACL (1)*, pages 1023–1032. Citeseer.
- Atish Pawar and Vijay Mago. 2018. Calculating the similarity between words and sentences using a lexical database and corpus statistics. *arXiv preprint arXiv:1802.05667*.
- Hiroya Takamura and Manabu Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics.
- Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181.
- Dingding Wang, Tao Li, Shenghuo Zhu, and Chris Ding. 2008. Multi-document summarization via sentence-level semantic analysis and symmetric matrix factorization. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM.
- Xun Wang, Masaaki Nishino, Tsutomu Hiraio, Katsuhito Sudoh, and Masaaki Nagata. 2016. Exploring text links for coherent multi-document summarization. pages 213–223.
- Markus Zopf, Maxime Peyrard, and Judith Eckle-Kohler. 2016. The next step for multi-document summarization: A heterogeneous multi-genre corpus built with a novel construction approach. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1535–1545.

Grafting Pre-trained Models for Multimodal Headline Generation

Lingfeng Qiao[†], Chen Wu[†], Ye Liu[†], Haoyuan Peng[†], Di Yin[†], Bo Ren[§]

[†]Tencent Youtu Lab, Shanghai, China

[§]Tencent Youtu Lab, Hefei, China

{leafqiao, rafelliu, haoyuanpeng, endymecyyin, timren}@tencent.com, overwindows@icloud.com

Abstract

Multimodal headline utilizes both video frames and transcripts to generate the natural language title of the videos. Due to a lack of large-scale, manually annotated data, the task of annotating grounded headlines for video is labor intensive and impractical. Previous researches on pre-trained language models and video-language models have achieved significant progress in related downstream tasks. However, none of them can be directly applied to multimodal headline architecture where we need both multimodal encoder and sentence decoder. A major challenge in simply gluing language model and video-language model is the modality balance, which is aimed at combining visual-language complementary abilities. In this paper, we propose a novel approach to graft the video encoder from the pre-trained video-language model on the generative pre-trained language model. We also present a consensus fusion mechanism for the integration of different components, via inter/intra modality relation. Empirically, experiments show that the grafted model achieves strong results on a brand-new dataset collected from real-world applications.

1 Introduction

In the age of information explosion, generating headlines of videos has been steadily gaining prominence on the short video platform. As the headlines can summarize the videos for people quickly acquiring their essential information. Good headlines are also beneficial for various scenarios, such as video retrieval, recommendation, and understanding (Zhu et al., 2022; Liu et al., 2022). Specially, video headline generation can be regarded as a textual generation task with multimodal inputs (Li et al., 2021), which is called multimodal generation as shown in Figure 1. Given a video with related transcript, algorithm aims to generate a short, concise and readable textual attraction title.

However, to build an effective model, collecting large-scale training data is the main challenge.

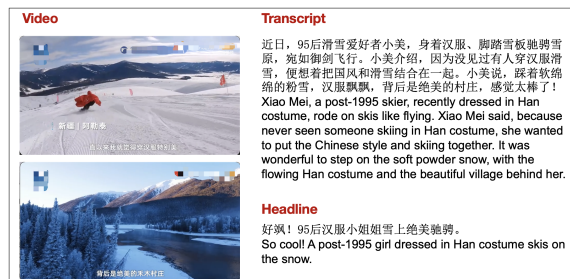


Figure 1: An example of multimodal generation.

Especially, in the multimodal headline generation task, the form of triplet data (video, source transcript, target summary) further increases the difficulty of collecting data, which limits the application of video headline generation.

To alleviate the issue in multimodal headline generation, the natural idea is to leverage pre-trained model (PTM). With large-scale corpus, such as GPT (Radford et al., 2018), BART (Lewis et al., 2019), PALM (Bi et al., 2020), etc., have shown great ability to generate readable and informative text. Since the multimodal headline generation combine both video and textual information, we propose that the model can be grafted by PTMs of language generation and video-text matching. The former provides the ability of headline generation, and the latter bridges the semantic gap between the multiple modalities (Radford et al., 2021). In addition, these two tasks have no concern of scarce data. For language generation, the existing pre-training model can be directly adopted. For video-text matching, the model can be trained with sufficient data from the Internet without manual annotations. By this means, multimodal headline generation model can be constructed by fine-tuning the grafted model with limited data collection.

In order to take advantage of the existing PTMs and improve reusability, we propose a **grafting** mechanism for obtaining the **multimodal summarization pre-training model** (GraMMo). First, language generation and video-text matching tasks are

introduced to pre-train the encoders and decoder, respectively. Then we construct a unified architecture with a video encoder, a text encoder and a text decoder which grafted from different PTMs to initialize the multimodal headline generation model. In addition, a joint-modality layer acted as a modality-balance gate is designed to fuse the video and text features. Unlike previous works which focus on retaining modality-shared feature (Libovický et al., 2018; Yu et al., 2021), this layer uses a two-way attention strategy to capture the commonality and specialty of the modalities. In detail, the video modality can highlight the most relevant and important text tokens by video-text cross attention. The resulted feature is called video-enhanced text feature, which reflect the commonality. On the other hand, since video-enhanced text feature neglects the video specialty which is less related with text modality, we recombine the video embeddings according to video-text attention and exploit the video-specific feature for complementing the fusion representation. Furthermore, dynamic frame sampling (DFS) and masked word prediction are designed in the encoder parts to reinforce the multimodal representation. We summarize the main contributions as follows:

- We propose a grafting video-text pre-training framework for multimodal headline generation. By grafting PTMs of language generation and video-text matching, GraMMo can be efficiently trained without big data collection. It is beneficial for fast deployment of real-world applications.
- A joint-modality layer with multimodal fusion module is designed to pay balanced attentions to each modality. It uses a two-way attention strategy to capture the commonality and specialty of multiple modalities, which will reinforce the fusion representation for better headline generation.
- Extensive experiments on a proposed Chinese multimodal headline generation dataset, WB-News, demonstrate that the grafted model can effectively accelerate the downstream fine-tuning procedure and improve generation results. The proposed method has been deployed in an industrial media platform for Chinese video headline generation.

2 Related Work

2.1 Multimodal Generation

Multimodal generation task aims to generate short, concise and readable textual title that can capture the most core information of the input media. The task is closely relevant to text summarization (Zhang et al., 2020; Jiang et al., 2022) while it is much tougher because redundancy and complementary between multiple modalities should be studied (Jangra et al., 2021). Many literatures have been based on pre-extracted unimodal sequential representation and cross attention mechanism to obtain the fused feature (Li et al., 2020d; Khullar and Arora, 2020; Li et al., 2020b; Fu et al., 2020). Besides, some researchers took efforts to sufficiently fusing the multimedia inputs by hierarchical fusion (Liu et al., 2020; Yu et al., 2021; Zhang et al., 2021a). The objective of modality consistency is another tool to guide the learning of multimodal fusion (Zhu et al., 2020; Zhang et al., 2021b).

However, few existing methods studied PTM for multimodal generation (Seo et al., 2022). In this paper, we propose a grafting video-text generation model and a novel joint-modality layer which is designed to capture the commonality and specialty of multiple modalities.

2.2 Video-Text Pre-training

Video-Text pre-training models adopt the "pre-training and then fine-tuning" paradigm, which makes the downstream tasks able to utilize the abundant knowledge included in pre-training data.

One class of work is task-specific pre-training, and contrastive learning is used for zero-shot transfer and video-text retrieval tasks, such as CLIP (Radford et al., 2021) and other related researches (Miech et al., 2019; Patrick et al., 2020; Huang et al., 2021; Xu et al., 2021b). Furthermore, CBT (Sun et al., 2019a), HERO (Li et al., 2020c), VideoAsMT (Korbar et al., 2020) and UniVL (Luo et al., 2020) adopt multi-task learning (MTL) for pre-training on retrieval tasks. The other class of work concentrates on how to interact the multimodal inputs, including VideoBERT (Sun et al., 2019b), Unicoder-VL (Li et al., 2020a), VL-BERT (Su et al., 2019), UNITER (Chen et al., 2020), VLP (Zhou et al., 2018), ActBERT (Zhu and Yang, 2020), VLM (Xu et al., 2021a) and BEiT (Wang et al., 2022).

Currently, few video-text pre-training models focus on multimodal headline generation due to the

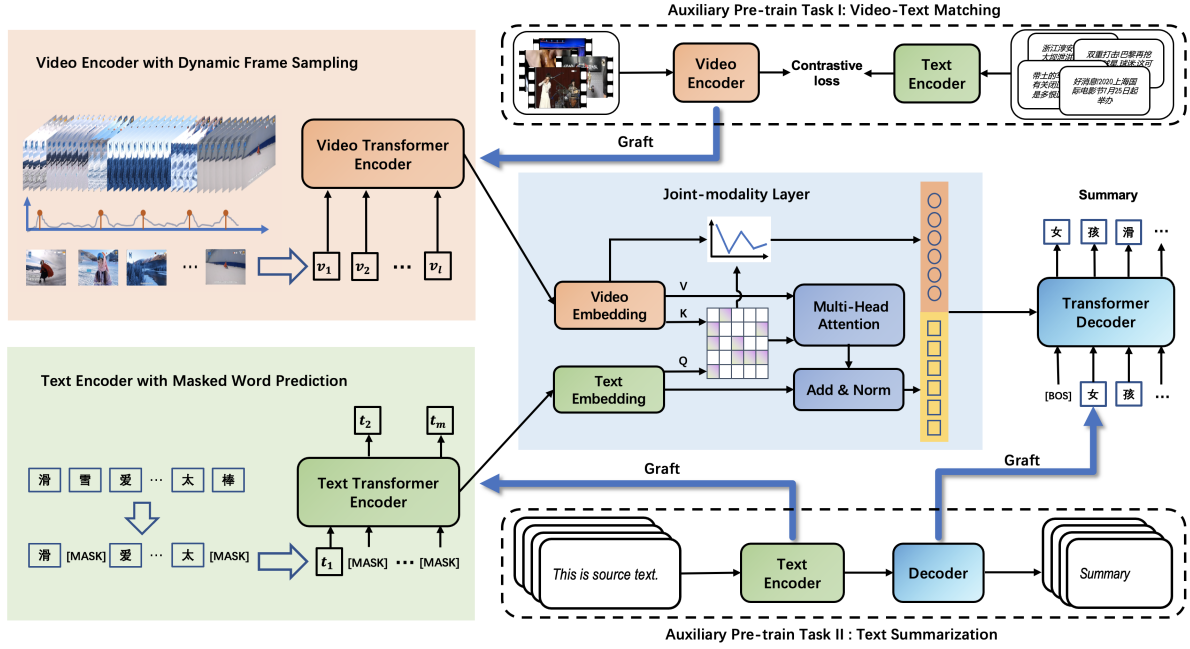


Figure 2: GraMMo framework for multimodal headline generation.

scarce data. GraMMo gives an effective grafting architecture for this task with ready-made PTMs of language generation and video-text matching, which can save a lot of computational costs.

3 Approach

3.1 Grafting Architecture

Given an input video $X_v = \{v_1, v_2, \dots, v_l\}$ and related source transcript $X_t = \{t_1, t_2, \dots, t_m\}$, the output is the target textual title $Y = \{y_1, y_2, \dots, y_n\}$, where l, m, n are the numbers of the corresponding tokens. The goal is to generate a predicted title $Y' = \{y'_1, y'_2, \dots, y'_n\}$ based on X_v and X_t , which can successfully grasp the main points of the video and transcript.

The proposed architecture is to provide a PTM for multimodal headline generation without large-scale triplet samples $\{X_v, X_t, Y\}$. The concept of grafting architecture GraMMo is illustrated in Figure 2. As a unified structure for multimodal generation, GraMMo consists of a video encoder $E_v(\cdot)$, a text encoder $E_t(\cdot)$, a joint-modality layer $F(\cdot)$ and a text decoder $D(\cdot)$. The encoders are designed for each modality individually and the architecture can be easily extended to various multimodal tasks with different kinds of inputs. Then the embeddings of modalities are fused by joint-modality layer to obtain the multimodal features. The joint-modality layer can provide video-enhanced text feature and video-specific feature, which involve the common-

ality and speciality of video and text modalities. Finally, text decoder is used to generate headline based on the fused multimodal feature.

To pre-train the video encoder, text encoder and text decoder, we draw support from two auxiliary tasks, i.e. language generation and video-text matching. As Figure 2 shows, the video encoder of video-text matching, the text encoder, and the text decoder of language generation are grafted to obtain the multimodal headline generation model.

3.2 Pre-train Generative Language Model

The language generation model with encoder-decoder structure can be adopted to build text encoder and decoder. In the work, PALM (Bi et al., 2020) which is Transformer-based (Vaswani et al., 2017) architecture is used as the NLG model. The text encoder and text decoder are pre-trained as classic abstractive text summarization task with large-scale unlabeled corpus.

In the pre-training stage, text encoder $E_t(\cdot)$ encodes source transcripts to obtain text embeddings $e_t = E_t(X_t)$, and then the decoder $D(\cdot)$ learns to generate hypothesis summaries. The generation loss and masked word prediction loss are used to guide the learning of text encoder and text decoder.

3.3 Pre-train Video-Text Understanding

We also use Transformer architecture for video encoder. The video features, extracted by I3D net-

work (Carreira and Zisserman, 2017), are first projected to video tokens before being fed into the video Transformer. For a video X_v , it has s frames, which is denoted as $V_f = \{f_1, f_2, \dots, f_s\}$. Due to the concern of model efficiency, the frames should be sampled and converted to the video tokens with l length. Most conventional methods used pre-extracted features based on uniform sampling from the raw video, e.g. extract one frame every one second duration of the video. However, this sampling method may limit the expression of video. In the framework, to enhance video embeddings, dynamic frame sampling (DFS) is designed. It is a projection layer $DFS(\cdot) : \{1, 2, \dots, l\} \rightarrow \{1, 2, \dots, s\}$, which represents the choice from variable frames.

As a result, the video tokens can be obtained by $v_i = f_{DFS(i)}$, where $i = 1, 2, \dots, l$. Then the I3D features of these tokens are extracted and video embeddings $e_v = E_v(X_v)$ are acquired by a stacked Transformer encoder.

To pre-train the video encoder $E_v(\cdot)$, video-text matching task is adopted to bridge the semantic gap between video and text modalities. Specifically, we collect large-scale video-text pairs from public video platform on the Internet without manual annotations. Videos and their corresponding descriptions are the natural data for video-text matching. The proposed video encoder and another Transformer-based text encoder are used to encode videos and descriptions respectively. Contrastive loss InfoNCE (Oord et al., 2018) is employed to calculate the correspondence between embeddings and guide the pre-training of encoders.

3.4 Joint-Modality Layer

Joint-modality layer will be used in fine-tuning stage after model grafting. Given text embeddings $e_t \in R^{m \times d}$ and video embeddings $e_v \in R^{l \times d}$, where d is the dimension of the embeddings, joint-modality layer is proposed to fuse them for headline generation. First, video embeddings should be used to highlight the significant elements in text embeddings and make algorithm pay attention to them from redundant source transcripts. Second, video embeddings can supplement key information that is not included in text embeddings to improve the informativeness of headline. To realize these motivations, as Figure 2 shown, the joint-modality layer uses a two-way attention strategy to capture the commonality and speciality of multiple modalities. Denote the query $Q \in R^{m \times d}$ is projected from

text embeddings e_t , and the key $K \in R^{l \times d}$ and value $V \in R^{l \times d}$ are projected from video embeddings e_v . With dot-product between Q and K , the video-text attention matrix $M_{vt} \in R^{m \times l}$ is obtained, which represents the relations between text and video tokens. On the one hand, the text features are enhanced by video information based on M_{vt} . Multi-head attention is applied and V is added to the related text tokens to obtain the video-enhanced text feature $g = e_t + M_{vt}V$.

On the other hand, video embeddings e_v can be divided into two aspects, i.e. text-relevant feature and video-specific feature. Text-relevant feature is the part of e_v with large video-text attention score and video-specific feature is the opposite part. The text-relevant feature has already been considered in $M_{vt}V$. On the contrary, the video-specific feature is neglected and should be supplemented. We calculate video-text relevant distribution $p \in R^{1 \times l}$ by summing M_{vt} along the query dimension. The lower value in p means that such a video embedding is less relevant to text, which should be chosen for video-specific feature. As a result, the video-specific feature h is obtained by $h = e_v \odot Norm(1 - p)$, where $Norm(\cdot)$ means the normalized operator and \odot is the element-wise multiplication.

Finally, the fusion embeddings $F(e_t, e_v)$ is obtained by concatenating video-enhanced text feature g and video-specific feature h .

3.5 Fine-tune Multimodal Generation Model

As mentioned above, text encoder $E_t(\cdot)$ and text decoder $D(\cdot)$ are pre-trained by language generation task. Video encoder $E_v(\cdot)$ is pre-trained by video-text matching task. By grafting these modules with joint-modality layer, a multi-modality generation model is established, which can be used for multimodal headline generation task.

For realistic applications, the specific multimodal headline generation triplet data should be collected to fine-tune the model. With GraMMo, we only need to prepare a small amount of data, since most of parameters in model are initialized by grafting, the model can converge rapidly.

4 Experiments

4.1 Datasets and Implementation

We leverage billions of Chinese corpus and millions of videos for pre-training language model and video-text matching models, respectively. For mul-

Methods	R-1	R-2	R-L	B-1	B-2	B-3	B-4	M
<i>Text</i> \rightarrow <i>Text</i>								
BART (Lewis et al., 2019)	33.14	19.51	29.41	32.54	25.34	19.60	15.39	29.48
PALM (Bi et al., 2020)	36.73	23.10	33.86	34.01	27.47	21.76	17.37	32.10
<i>Video</i> \rightarrow <i>Text</i>								
VLM (Xu et al., 2021a)	5.10	0.61	4.44	4.31	1.54	0.64	0.29	2.89
GraMMo-Video	7.85	1.73	6.90	7.21	3.31	1.76	1.01	5.06
<i>Video+Text</i> \rightarrow <i>Text</i>								
VG-GPLM (Yu et al., 2021)	35.35	21.46	32.10	33.81	26.70	20.78	16.38	31.01
MV-GPT (Seo et al., 2022)	37.74	24.04	34.42	36.13	29.27	23.34	18.81	33.73
MMPT (Xu et al., 2021b)	38.21	24.25	35.05	31.65	25.77	20.58	16.65	31.76
GraMMo	38.87	24.85	35.38	37.80	30.65	24.43	19.65	35.30

Table 1: Headline generation results on WB-News dataset.

timodal headline generation fine-tuning and testing, a new dataset WB-News is built. The details of datasets and the methodology used to obtain the corpus can be found in supplementary materials A. For evaluation on WB-News dataset, three metrics are employed: ROUGE (R-1,R-2,R-L) (ROUGE, 2004), BLEU (B-1,B-2,B-3,B-4) (Papineni et al., 2002), METEOR (M) (Banerjee and Lavie, 2005).

4.2 Headline Generation Results

On WB-News dataset, according to different types of input, we compare generation methods in three kinds of experimental setups, i.e. *Text* \rightarrow *Text*, *Video* \rightarrow *Text* and *Video+Text* \rightarrow *Text*. **BART** and **PALM** are used as baselines for classic text-only generation. Without text modality, **GraMMo-Video** is designed by pruning text encoder in GraMMo, which is compared with the video caption method **VLM**. For multimodal generation task, latest methods **VG-GPLM**, **MV-GPT** and **MMPT** are compared with the proposed **GraMMo**. As Table 1 shows, GraMMo achieves the best results among related multimodal generation methods and the SOTA text summarization method, PALM, with large margin. It illustrates that the video modality can help text to improve the headline results and GraMMo can better leverage the multimodal information against the related methods.

In the *Video* \rightarrow *Text* scenario, because the factual information, such as the name, cannot be extracted using the video modality, the headline metrics are quite low. Nevertheless, **GraMMo-Video** achieves a better performance and can generate reasonable summaries if neglecting the factual consistency. It also shows that our method has the ability to extract the language semantics from video.

Furthermore, GraMMo has been deployed on an industrial platform for video headline generation, which is shown in the discussion section.

Pre-trained Models		R-L	B-4	M
Text	Video			
		26.98	12.18	24.65
	✓	27.63	12.30	24.69
✓		34.90	18.63	34.06
✓	✓	35.38	19.65	35.30

Table 2: The effectiveness of using grafted model.

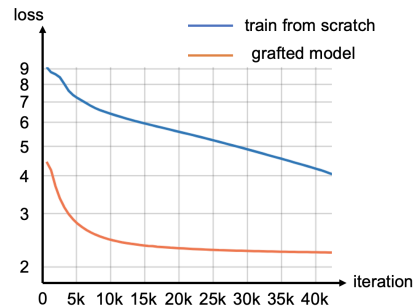


Figure 3: Different initialization methods.

4.3 Grafting for Headline Generation

In Table 2, the generation results are compared by eliminating grafted components. We found that the performance can be significantly improved by grafting. Figure 3 also shows that the learning curve of fine-tuning with grafted model decreases and converges more rapidly against the case of train-from-scratch.

4.4 Ablation Study

To verify the contributions of each component of GraMMo, we design a series of ablation experiments. The results are shown in Table 3. First, all summary metrics decrease when DFS and video Transformer are removed from the video encoder. The probable reasons are that DFS can improve the generalization of video embedding and video Transformer can model the sequential character of video tokens. Second, different fusion strate-

Ground Truth	GraMMo (Video + Text → Text)	PALM (Text → Text)	GraMMo-Video (Video → Text)
好飒！95后汉服小姐姐雪上绝美驰骋 So cool! A post-1995 girl dressed in Han costume skis on the snow .	95后滑雪爱好者雪中宛如御剑飞行 A post-1995 skier skis like flying on the snow .	95后滑雪爱好者雪中觅食 A post-1995 skier foraged for food in the snow.	东北人在哪里玩雪？一起来解一下冬奥会滑雪 Where do the northeast people play snow? Let's take a look at winter Olympic skiing.
儿子一眼认出奥特曼是爸爸假扮：我不喜欢大肚子奥特曼 Boy recognized at a glance the Ultraman is the impersonation of his father. 'I don't like big belly Ultraman '.	爸爸给儿子制造惊喜奥特曼敲门，儿子：我不喜欢大肚子，我知道是爸爸 Father dressed in Ultraman and knocked at the door to make surprise for his son. Boy: 'I don't like big belly . He is my dad.'	爸爸为给儿子庆生制造惊喜，大肚子我知道是爸爸 Father make surprise for his son's birthday. ' Big belly . I know he is my dad.'	可可爱爱！小女孩给消防员送苹果 So lovely! A little girl gave the fireman apples.
幸福时刻！载人航天发射塔架见证航天人婚礼 Happy moment! man-carrying rocket launch tower witness the wedding of aerospace industry staff .	酒泉卫星发射中心为120对航天新人举行婚礼 Jiuquan Satellite Launch Center held a wedding ceremony for 120 newlyweds , which work for aerospace industry .	甜甜祝福！神舟飞船启航新人们集体婚礼 Sweet blessing! The Shenzhou spacecraft set off and the newlyweds group wedding .	向全国各族人民致以新春祝福 Happy New Year greetings to the people of the whole country.

Figure 4: Case study of proposed GraMMo, PALM and GraMMo-Video.

Methods	R-L	B-4	M
GraMMo	35.38	19.65	35.30
<i>Encoder</i>			
w/o DFS	-0.28	-0.13	-0.53
w/o Video Transformer	-0.98	-0.26	-1.03
<i>Fusion</i>			
Naive Concat	-0.89	-0.96	-1.40
Cross Attention	-0.85	-0.66	-1.16

Table 3: Ablation study on model components.

gies are studied by substituting joint-modality layer. Naive concatenate and cross attention are adopted and decrease the summarization performance to great extent. The phenomenon illustrates that the proposed joint-modality layer is more effective in utilizing the multimodal information.

5 Discussion

5.1 Video-Text Matching Helps

One key issue of multimodal headline generation is how to map the video and text into the joint embedding space. Therefore, the alignment of these two modalities is important, which can be reflected by video-text retrieval performance.

We selected 1,400 samples of video and its title pair from WB-News to conduct video-text retrieval experiments. The video embeddings e_v and text embeddings e_t extracted from video encoder $E_v(\cdot)$ and text encoder $E_t(\cdot)$ are computed by dot product to measure the relevant scores. Given a query, the most related items are recalled by sorting the scores. Recall metrics (R@1, R@5, R@10) are used to measure the results. As shown in Table 4, the R@1 achieves about 40% and R@5 about 60%, which means the video and text are well aligned in one common space. Moreover, when using grafted model, the retrieval results are better than the results of train-from-scratch, reflecting the superiority of the grafted model.

Methods	R@1	R@5	R@10
<i>Train from Scratch</i>			
text-to-video	34.40	53.19	60.28
video-to-text	35.39	52.70	60.21
<i>Grafted Model</i>			
text-to-video	39.65	60.43	67.73
video-to-text	40.35	60.35	67.30

Table 4: Video-Text Retrieval Results

5.2 Complementary Relation

Several examples are provided to intuitively understand the effectiveness of modality balance. As shown in Figure 4, the generated hypotheses of GraMMo, PALM and GraMMo-Video are presented. The key information is emphasised by green words, while the red words mean the wrong predictions. Compared with PALM, GraMMo can extract more key information and produce more logical expressions. This effect demonstrates the value of video modality for generating more remarkable headline.

However, the generated hypotheses of GraMMo-Video are totally inconsistent with the ground-truths because the factual information cannot be extracted solely by video modality. In fact, GraMMo-Video can generate the words containing similar topic semantics, which means our method can establish the connection between video and text modalities.

5.3 Human Evaluation

We perform human evaluation from the perspectives of readability and informativeness. For all test samples, the source video, reference headlines, and generated headline are shown to a group of people for evaluation. They need to judge the two aspects of readability and informativeness by giving an integer score in the range of 1-5, with 5 being perfect. Each sample is assessed by 5 people, and the average scores are used as the final score.

Methods	Read.	Info.
Ground Truth	4.35	4.05
PALM	3.65	3.08
MMPT	3.70	3.3
GraMMo	3.71	3.54

Table 5: Human evaluation results on readability (Read.) and informativeness (Info.) of generated headlines.

As shown in Table 5, we find that the GraMMo performs better readability and informativeness scores compared with PALM and MMPT, demonstrating its effectiveness in generating informative headlines. For readability, all the three headline generating methods can generate quite readable language. This is because a large training corpus can make text decoder generate coherent sentences, except for the mistakes of repetition phrases and grammatical errors. For informativeness, the major problems are the fact inconsistency and incomplete key information. They will be investigated in future work to improve the quality of generated headlines.

5.4 Media AI Platform

Our Chinese video headline generation is deployed in an AI platform for industrial media, which is a well-designed video understanding platform with complete video processing services. When generating headline, GraMMo takes the source video and its pre-extracted ASR text as the inputs and then predicts the textual summary as the headline of the video. We give some headline generation examples of real Chinese news videos, as shown in Figure 5.

6 Conclusion

In this paper, we propose GraMMo, grafting a pre-trained sequence-to-sequence language model and a video-language understanding model for multimodal headline generation. By fine-tuning the representation components (video-encoder&text-encoder) and generation component (text-decoder) of the model, we alleviate the problem of lacking large-scale dataset in multimodal headline generation. To capture the commonality and specialty of the video and text features, we propose an extra fusion layer to balance modalities and maximally maintain the original architectures. With this approach, we can fully take advantage of the pre-trained models, including well-trained capacity for multimodal understanding and generation. Experiments results show that our method can significantly improve the performance and outperform



Figure 5: The examples of video headline generation results on news videos in the media AI platform. Red text boxes illustrate the generated titles based on the video and ASR text information.

similar works. Furthermore, the proposed method has also been applied effectively and efficiently in our online system. We will release the WB-News dataset, GraMMo code, and the grafted models.

7 Ethics Considerations

The authors declare that the use of data in our research is permitted. First, the Chinese corpus used in the text summarization auxiliary task is an open-source dataset. Second, for video-text data used in our multimodal headline generation, the data are collected and used in accordance with the privacy policies of short video platforms.

Ethical concerns include the usage of the proposed model for a purpose directly different from the previously mentioned headline generation task, such as hateful memes generation by feeding irrelevant video and text inputs, as well as integration in public opinion manipulation tools.

References

- Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.
- Bin Bi, Chenliang Li, Chen Wu, Ming Yan, Wei Wang, Songfang Huang, Fei Huang, and Luo Si. 2020. Palm: Pre-training an autoencoding&autoregressive language model for context-conditioned generation. *arXiv preprint arXiv:2004.07159*.
- Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6299–6308.
- 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 *European conference on computer vision*, pages 104–120. Springer.
- Xiyan Fu, Jun Wang, and Zhenglu Yang. 2020. Multimodal summarization for video-containing documents. *arXiv preprint arXiv:2009.08018*.
- Po-Yao Huang, Mandela Patrick, Junjie Hu, Graham Neubig, Florian Metze, and Alexander Hauptmann. 2021. Multilingual multimodal pre-training for zero-shot cross-lingual transfer of vision-language models. *arXiv preprint arXiv:2103.08849*.
- Anubhav Jangra, Adam Jatowt, Sriparna Saha, and Mohammad Hasanuzzaman. 2021. A survey on multi-modal summarization. *arXiv preprint arXiv:2109.05199*.
- Zhuoxuan Jiang, Lingfeng Qiao, Di Yin, Shanshan Feng, and Bo Ren. 2022. Leveraging key information modeling to improve less-data constrained news headline generation via duality fine-tuning. *arXiv preprint arXiv:2210.04473*.
- Aman Khullar and Udit Arora. 2020. Mast: Multimodal abstractive summarization with trimodal hierarchical attention. *arXiv preprint arXiv:2010.08021*.
- Bruno Korbar, Fabio Petroni, Rohit Girdhar, and Lorenzo Torresani. 2020. Video understanding as machine translation. *arXiv preprint arXiv:2006.07203*.
- 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*.
- Gen Li, Nan Duan, Yuejian Fang, Ming Gong, and Daxin Jiang. 2020a. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11336–11344.
- Haoran Li, Junnan Zhu, Jiajun Zhang, Xiaodong He, and Chengqing Zong. 2020b. Multimodal sentence summarization via multimodal selective encoding. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5655–5667.
- Junyi Li, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Pretrained language models for text generation: A survey. *arXiv preprint arXiv:2105.10311*.
- Linjie Li, Yen-Chun Chen, Yu Cheng, Zhe Gan, Licheng Yu, and Jingjing Liu. 2020c. Hero: Hierarchical encoder for video+ language omni-representation pre-training. *arXiv preprint arXiv:2005.00200*.
- Mingzhe Li, Xiuying Chen, Shen Gao, Zhangming Chan, Dongyan Zhao, and Rui Yan. 2020d. Vmsmo: Learning to generate multimodal summary for video-based news articles. *arXiv preprint arXiv:2010.05406*.
- Jindrich Libovický, Shruti Palaskar, Spandana Gella, and Florian Metze. 2018. Multimodal abstractive summarization of open-domain videos. In *Proceedings of the Workshop on Visually Grounded Interaction and Language (ViGIL). NIPS*.
- Nayu Liu, Xian Sun, Hongfeng Yu, Wenkai Zhang, and Guangluan Xu. 2020. Multistage fusion with forget gate for multimodal summarization in open-domain videos. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1834–1845.
- Ye Liu, Lingfeng Qiao, Di Yin, Zhuoxuan Jiang, Xinghua Jiang, Deqiang Jiang, and Bo Ren. 2022. Os-msl: One stage multimodal sequential link framework for scene segmentation and classification. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 6269–6277.
- Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming Zhou. 2020. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*.
- Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- 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*, pages 311–318.

- Mandela Patrick, Po-Yao Huang, Yuki Asano, Florian Metze, Alexander Hauptmann, Joao Henriques, and Andrea Vedaldi. 2020. Support-set bottlenecks for video-text representation learning. *arXiv preprint arXiv:2010.02824*.
- 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. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.
- Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.
- Lin CY ROUGE. 2004. A package for automatic evaluation of summaries. In *Proceedings of Workshop on Text Summarization of ACL, Spain*.
- Paul Hongsuck Seo, Arsha Nagrani, Anurag Arnab, and Cordelia Schmid. 2022. End-to-end generative pre-training for multimodal video captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17959–17968.
- Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2019. Vi-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*.
- Chen Sun, Fabien Baradel, Kevin Murphy, and Cordelia Schmid. 2019a. Learning video representations using contrastive bidirectional transformer. *arXiv preprint arXiv:1906.05743*.
- Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. 2019b. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7464–7473.
- 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.
- Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. 2022. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. *arXiv preprint arXiv:2208.10442*.
- Hu Xu, Gargi Ghosh, Po-Yao Huang, Prahal Arora, Masoumeh Aminzadeh, Christoph Feichtenhofer, Florian Metze, and Luke Zettlemoyer. 2021a. Vlm: Task-agnostic video-language model pre-training for video understanding. *arXiv preprint arXiv:2105.09996*.
- Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. 2021b. Video-clip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*.
- Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, et al. 2020. Clue: A chinese language understanding evaluation benchmark. *arXiv preprint arXiv:2004.05986*.
- Tiezheng Yu, Wenliang Dai, Zihan Liu, and Pascale Fung. 2021. Vision guided generative pre-trained language models for multimodal abstractive summarization. *arXiv preprint arXiv:2109.02401*.
- 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.
- Litian Zhang, Xiaoming Zhang, Junshu Pan, and Feiran Huang. 2021a. Hierarchical cross-modality semantic correlation learning model for multimodal summarization. *arXiv preprint arXiv:2112.12072*.
- Zhengkun Zhang, Xiaojun Meng, Yasheng Wang, Xin Jiang, Qun Liu, and Zhenglu Yang. 2021b. Unims: A unified framework for multimodal summarization with knowledge distillation. *arXiv preprint arXiv:2109.05812*.
- Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. 2018. End-to-end dense video captioning with masked transformer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8739–8748.
- Junnan Zhu, Yu Zhou, Jiajun Zhang, Haoran Li, Chengqing Zong, and Changliang Li. 2020. Multimodal summarization with guidance of multimodal reference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9749–9756.
- Linchao Zhu and Yi Yang. 2020. Actbert: Learning global-local video-text representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8746–8755.
- Peng Zhu, Dawei Cheng, Siqiang Luo, Ruyao Xu, Yuqi Liang, and Yifeng Luo. 2022. Leveraging enterprise knowledge graph to infer web events’ influences via self-supervised learning. *Journal of Web Semantics*, page 100722.

A Experimental details

A.1 Dataset

The datasets for pre-training and fine-tuning are listed as follows.

A.1.1 Pre-training for NLG

We leverage 14GB high quality Chinese corpus of CLUE-small¹ (Xu et al., 2020). It contains following genres:

News This sub-corpus is crawled from the We Media (self-media) platform, with a total of 3 billion Chinese words from 2.5 million news articles from roughly 63K sources.

WebText With 4.1 million questions and answers, the WebText sub-corpus is crawled from Chinese Reddit-like websites such as Wukong QA, Zhihu, Sogou Wenwen, etc.

Wikipedia This sub-corpus is gathered from the Chinese content on Wikipedia (Chinese Wikipedia), containing around 1.1 GB of raw texts with 0.4 billion Chinese words on a wide range of topics.

Comments These comments are collected from E-commerce websites including Dianping.com and Amazon.com by SophonPlus². This subset has approximately 2.3 GB of raw texts with 0.8 billion Chinese words.

A.1.2 Pre-training for Video-Text Matching

We collect 3.2 million videos from the Chinese video platform. The topics of video cover news, sports, entertainments, etc. For each video, its headline information is edited by uploader so that the video-text matching task can be conducted without additional manual annotations.

A.1.3 Multimodal Generation Fine-tuning

We establish the WB-News dataset for multimodal generation fine-tuning and evaluation. It contains more than 43,000 samples that are collected from official Weibo accounts of China’s main media.

When building WB-News, we first filter the weibo contents which have corresponding videos. Then, the raw data is manually annotated and cleaned to produce the triplet form, i.e. video, source transcript and target summary. The average video duration of these samples is about one minute, the average length of transcript is 120.6

words and the average length of summary is 21.2 words. For testing, 697 samples are selected, which can evaluate the performance of Chinese video headline generation.

A.2 Hyper-parameters

Text Encoder We use PALM as the text pre-training model, in which 6-layer Transformers are used for both text encoder and decoder. The text tokens of samples are padded to 128 lengths.

Video Encoder For the video encoder, DFS extracts 32 video tokens from each video. Then a 2-layer Transformer encoder with 8 attention heads is applied to get video embedding.

Joint-modality Layer After obtaining the outputs encoded by text and video Transformers, linear projection layers are used to project them into the same 512 dimension. As a result, the feature dimension of $e_v \in R^{32 \times 512}$ and $e_t \in R^{128 \times 512}$. Then a video-text attention matrix M_{vt} with 8 heads is established to produce the fusion embeddings.

Decoder In the decoding stage, we use beam search with a beam size of 5. The decoding process will not stop until an end-of-sequence (EOS) token is emitted or the length of the generated summary reaches to 64 tokens.

Training Details GraMMo is realized by fairseq toolkit³. In training phrase, we use learning rates 3e-4 to pre-train the encoders and decoder, and 1e-5 to fine-tune the model. Batch size is set to 64 and dropout rate is 0.1. Adam optimizer is adopted with 0.01 weight-decay and 0.1 clip-norm. The training procedure runs on 8 NVIDIA V100 GPU cards and costs about 7 days for NLG pre-training, 5 days for video-text matching pre-training and 1 hour for multimodal generation fine-tuning.

¹<https://www.cluebenchmarks.com/>

²<https://github.com/SophonPlus/ChineseNlpCorpus/>

³<https://github.com/pytorch/fairseq>

Semi-supervised Adversarial Text Generation based on Seq2Seq models

Hieu Le*

Boston University, Boston, USA
hle@bu.edu

Dieu-Thu Le

Amazon Inc., Berlin, Germany
deule@amazon.com

Verena Weber

Amazon Inc., Berlin, Germany
wverena@amazon.com

Chris Church

Amazon Inc., Berlin, Germany
cbchurch@amazon.com

Kay Rottmann

Amazon Inc., Berlin, Germany
krrotm@amazon.com

Melanie Bradford

Amazon Inc., Berlin, Germany
neunerm@amazon.com

Peter Chin

Boston University, Boston, USA
spchin@bu.edu

Abstract

To improve deep learning models' robustness, adversarial training has been frequently used in computer vision with satisfying results. However, adversarial perturbation on text have turned out to be more challenging due to the discrete nature of text. The generated adversarial text might not sound natural or does not preserve semantics, which is the key for real world applications where text classification is based on semantic meaning. In this paper, we describe a new way for generating adversarial samples by using pseudo-labeled in-domain text data to train a seq2seq model for adversarial generation and combine it with paraphrase detection. We showcase the benefit of our approach for a real-world Natural Language Understanding (NLU) task, which maps a user's request to an intent. Furthermore, we experiment with gradient-based training for the NLU task and try using token importance scores to guide the adversarial text generation. We show that our approach can generate realistic and relevant adversarial samples compared to other state-of-the-art adversarial training methods. Applying adversarial training using these generated samples helps the NLU model to recover up to 70% of these types of errors and makes the model more robust, especially in the tail distribution in a large scale real world application.

*The work is completed during Hieu Le's internship at Amazon Inc.

1 Introduction

Over the past years, neural machine learning models have become more popular in a wide range of real world natural language applications. While these models have become very accurate, they are susceptible to errors due to small changes in the input, e.g. adding a stop word. This makes it difficult for a natural language understanding system to recognize all utterances correctly since there are many ways to formulate one request. In fact, while common user commands can be understood very well by the system, a system can react differently due to minor input changes, e.g., article variations, paraphrasing, or adding functional words.

Natural Language Understanding (NLU) is at the core of voice assistants and maps a user's request (also referred to as utterance later) to a specific intent within a certain domain, i.e. *PlayMusic* intent within the *Music* domain. In this paper, we identify the weaknesses of an NLU text classification model by finding the types of inputs that have high probability of causing prediction errors and show how to mitigate them. Inspired by recent work in adversarial training and adversarial sample generation (Goodfellow et al., 2015; Morris et al., 2020a), we describe how we employ adversarial text perturbation to identify and fix such samples to ensure a stable behavior of the model and thereby significantly boost the tail accuracy in a large scale real world application through adversarial training.

We show that while common text perturbation methods relying on character manipulation (Ebrahimi et al., 2018; Li et al., 2019), word swap (Alzantot et al., 2018; Ren et al., 2019) could generate adversarial samples on the original text inputs to trick the model, most of them often generate many irrelevant samples that either do not make sense (e.g., grammatically incorrect, unnatural, out of domain) or change the meaning of the original text utterance. Therefore, applying these methods in a real world application without any domain adaptation and additional constraints is not only unsuitable but could even harm the model. In this work, we describe how we use pseudo-labeled in-domain text data for task adaptation and thereby create adversarial perturbations that are more natural and relevant to the NLU task. Our approach uses in-domain data to train a seq2seq model that generates an adversarial version of the input utterance. We introduce multiple constraints including a paraphrase detector model to make sure that the generated adversarial text samples are of high quality. Finally, we show that including these generated adversarial samples during training helps to improve the model’s robustness. We compare the results with adversarial training based on the Fast Gradient Sign Method (FGSM) (Shafahi et al., 2019) often used in computer vision.

2 Related work

Adversarial sample generation in the text domain can be categorized by the level of the attempt towards a target sentence. At the lowest level, hotflip (Ebrahimi et al., 2018) and TextBugger (Li et al., 2019) perform character level perturbation by character manipulations (swap, insert or delete). Other methods by Alzantot et al. (2018), Zang et al. (2020) and Li et al. (2020) create attempts at word level using synonym swaps or masked language model. The most broad type of attempt is sentence level attempt where multiple words within a sentence are changed, such as PAWS (Zhang et al., 2019), SCPN (Iyyer et al., 2018) and SEARs (Ribeiro et al., 2018). While these methods can sometimes produce very high attempt success rates, they are not straightforward to apply out-of-the-box to real world applications, especially when the domains are specific and small changes in the original inputs can lead to a valid change in the labels (i.e., invalid attempts). In this work, we mainly deal with the problem of generating in-domain valid ad-

versarial attempts by using a seq2seq model that learns from live traffic/un-annotated data in real world applications.

3 Seq2Seq-based adversarial text perturbation

Figure 1 describes our semi-supervised seq2seq-based adversarial text perturbation (SSAT) framework which consists of three components: (1) adversarial candidate extraction from unannotated data based on pseudo-labels, (2) a paraphrase detector model that keeps only candidates that are actual paraphrases, and (3) a seq2seq (T5) model that is trained on the data generated in the first two steps to generate adversarial samples for a given input text.

3.1 Pseudo-label based data filtering

In industry applications, it is common that there is significantly more unlabeled data than high quality labeled data. Our goal is to leverage the unannotated data to find pairs of utterances that belong to the same class but are classified differently by the target model. To that end, we propose a multi-step funnel. First, we use the target classification model to pseudo-label the unlabeled data. Then, to obtain sentence vectors, the utterances are encoded using S-BERT (Reimers and Gurevych, 2019). We identify the k nearest neighbors in the embedding space for each sentence vector to obtain a dictionary with the $top k$ closest utterances in the embedding space for every utterance. Using the list of nearest neighbors for each utterance, we create k pairs and only keep those where the pseudo-label differs.

3.2 Paraphrase detection

We train a paraphrase detection model that determines if two sentences are paraphrases to further filter the dataset created in the previous step. We use a BERT model finetuned on the binary classification task. The training data for this task is extracted from labeled data where we used two different types of data. If a dataset contains only labels (e.g. intent) without specific name entity annotations, we ignore the name entities and define two sentences as paraphrases when they share the same label : $\text{PARAPHRASE}(x, x') : 1 \text{ if } y == y' \text{ else } 0$. If, however, the dataset contains labels (e.g. domain/intent) and slot labels (e.g. Named Entities), we define two sentences as paraphrases if they share the same non-absent named entities and in-

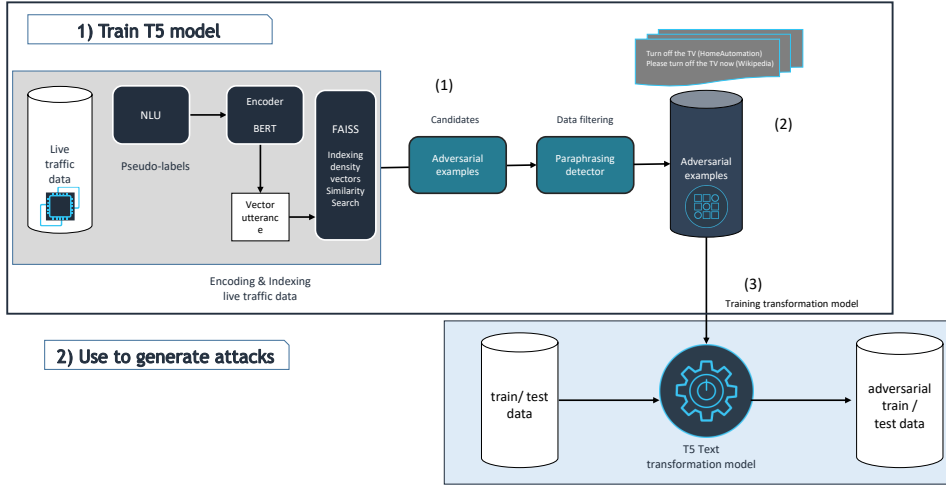


Figure 1: Components of our semi-supervised seq2seq-based adversarial text perturbation framework (SSAT): (1) a text filtering that finds texts that are similar by L_2 distance, (2) paraphrase detector model that detects and keeps only pairs that are paraphrases, (3) text perturbation recipe consisting of a seq2seq model trained on data created from (2).

tent/class. The detailed definition of labeled paraphrases is shown in Algorithm 1.

The pairs of semantically similar utterances but with different pseudo-labels from 3.1 are then fed into the paraphrase detector model. We only keep the the pairs classified as paraphrases and use them as training data for the next step.

```

Input :  $x = ([w_1|NE_1, \dots, w_n|NE_n], y)$ 
SLOT( $x$ ):
   $s = \emptyset$ 
  for  $w_i|NE_i \in x$  do
    if  $NE_i$  exists then
      add  $w_i|NE_i$  to  $s$ 
    end
  end
  return  $s$ 

PARAPHRASE( $x, x'$ ):
  if SLOT( $x$ ) == SLOT( $x'$ ) and  $y == y'$  then
    return True
  else
    return False
  end

```

Algorithm 1: Definition of Paraphrases

3.3 Text perturbation recipe

From the text filtering and paraphrase detection steps, we obtain data that satisfy multiple criteria of adversaries: **Semantic similarity**: ensured by knn search in BERT embedding space. **Meaning preservation**: ensured by a paraphrase detector model trained on labeled data with a dataset-specific definition of paraphrases. Intuitively, filtering the data with a paraphrase detector has several

advantages: a high level of fluency as the data come from real user traffic and a flexible constraint of semantic similarity that is not enforced through semantic agnostic measurements like edit distance or word embedding similarity.

Our perturbation recipe contains a text-to-text transformer model that creates variations of the input text, serving as candidates for testing the model. We use a pretrained Text-to-Text Transfer Transformer model (T5) (Raffel et al., 2020) fine-tuned on the filtered data for our task. The fine-tuned T5 model is then used to generate multiple adversarial candidates. Specifically, given an input x , we use T5 with a large beam to generate a fixed number of successful perturbation candidates. The heuristic used for beam search is cosine similarity to encourage higher similarity to the input text. From the set of successful candidates $T5(x) = [x_1, \dots, x_n]$, the candidate with the highest cosine similarity is returned. In terms of perturbation constraints, we reuse the paraphrase detector and thus, any returned perturbation candidate that switches class from the original text and is classified as a paraphrase of the input text is deemed a successful attempt. Figure 2 shows an example of adversarial text generation with T5 and beam search. In this example, from the original SNIPS input of "add this song to blues roots" (labeled *AddToPlaylist*), beam search generates multiple successful candidates ['play music from the playlist late night blues', 'add this tune to my blues playlist', ...]. The candidate 'add this tune to my blues playlist', wrongly classified as

PlayMusic, is the closest to the original input by cosine similarity and also satisfies the paraphrase constraint. Thus it is returned as a successful attempt.

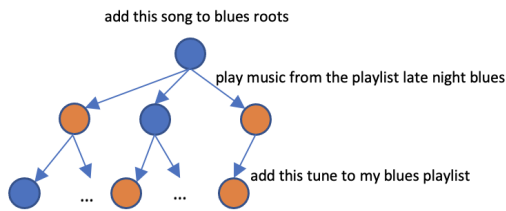


Figure 2: Adversarial text generation process on SNIPS example using beam search on T5 output. Blue nodes are text variants still correctly classified, orange nodes are variants that cause the target model to switch label.

3.4 Adversarial training with text perturbation

Besides finding adversarial perturbations that exploit model weaknesses, we also explored the usage of these generated utterances as additional training data to make the target model more robust against these types of adversarial utterances. We applied the recipe described in 3.3 to our regular training, evaluation and test data. From the generated utterances we added the successful perturbations, those identified as not altering meaning but switching the predicted class, as additional training data into the regular training process. For the test set we kept the original test set and the generated adversarial test set separate to also get insights into the effectiveness of the text perturbations and the effectiveness of the adversarial training.

4 Experiments

4.1 Dataset

We experiment with four text data sets, two commercial data sets from a virtual assistant in German and French, the SNIPS dataset (Coucke et al., 2018) and the MASSIVE data set (FitzGerald et al., 2022). The data consists of unannotated data as well as annotated training and test data. With the commercial data set, unannotated data, i.e. live traffic, is organic audio data coming in from users and processed through an acoustic speech recognition (ASR) system to convert it to text. In this commercial dataset, all data has been preprocessed and anonymized so that no user related information is identifiable. For SNIPS data set, we split data the same way as (Goo et al., 2018) for validation and

test set. However, we further split the training data set of 13,084 utterances into roughly 20 – 80 of labeled and ‘live traffic’ (unannotated) utterances: we remove the label for 11,000 utterances to simulate live traffic and keep 2,084 utterances annotated for training. Lastly, for MASSIVE data set, we use the training portion of the data set as the live-traffic, unannotated data set and use the validation and test data set to train the target model. Finally, the live-traffic portion of all the datasets are pseudo-labeled by our corresponding target models.

4.2 Implementation

For each dataset, we fine-tuned a pre-trained BERT architecture for text classification which all achieve above 90% accuracy. Those models serve as target models for the generated perturbations, i. e. we want to fool them through slight changes in the inputs.

For the paraphrase classifier, we chose a multilingual T5 (mT5) (Xue et al., 2021) model as it accommodates the considered datasets in different languages. The training data for the paraphrase classifier is processed from the training set by picking out positive and negative pairs. The positive pairs of paraphrases are picked based on definition 1. For negative examples, as the NLU dataset contains named entity slots, beside randomly selecting pairs of sentences with different classes, we also include cross-class pairs that share at least one slot. The binary classifier model is trained for 5 epochs with batch size of 16 and achieves an accuracy above 90% for both datasets.

We leverage the trained target models and the trained paraphraser to create a training dataset for our T5 model. First, we run the target model on the *live traffic* to acquire pseudo-labels. Using the pseudo-labels, utterances are sorted into buckets for each class. Then, for each utterance, we find the *knn* closest utterances in other buckets using FAISS (Johnson et al., 2021) similarity search. The detailed algorithm is shown in algorithm 2. By prefiltering the live traffic in this manner, we generate pairs of sentences that are close by L_2 distance yet classified differently by the target model. The candidate pairs are then passed through the trained paraphrase detector and all the positive pairs are kept.

4.3 Baselines

We compare our perturbation recipe with four other text perturbation recipes across different pertur-

		DeepWordBug	BAE	TextBugger	TextFooler	SSAT
German Commercial dataset	Success rate	92.38%	66.93%	64.14%	43.44%	61.93%
	Success paraphrase rate	27.59%	14.60%	27.25%	9.88%	46.69%
	Perplexity	1704.81	758.85	1239.13	857.00	498.25
	Average grammar issues	2.80	2.215	2.88	2.53	2.21
French Commercial dataset	Success rate	94.17%	78.35%	38.45%	70.54%	86.37%
	Success paraphrase rate	30.48%	28.23%	21.59%	25.18%	45.12%
	Perplexity	7087.31	2743.82	2820.99	4323.48	1712.14
	Average grammar issues	2.50	1.75	2.38	2.17	1.65
SNIPS	Success rate	57.98%	51.1%	22.59%	73.94%	90.42%
	Success paraphrase rate	49.14%	46.85%	11.14%	65.71%	12%
	Perplexity	294.06	82.90	139.42	159.59	6.3
	Average grammar issues	3.67	1.84	2.88	2.30	1.57
MASSIVE	Success rate	70.07%	67.32%	48.52%	18.09%	84.73%
	Success paraphrase rate	69.48%	66.79%	48.20%	18.03%	84.08%
	Perplexity	958.18	794.44	980.37	872.93	677.51

Table 1: All metrics for generated text perturbations across 4 baselines compared with SSAT: Success rate, success rate under paraphrase constraint, perplexity and average grammatical issues of generated perturbations. The threshold confidence level for paraphrased positive pairs constraint is set at 95%. Perplexity is calculated with *german-gpt2*, *gpt2-french-small*, *gpt2* and *mgpt* accordingly for the corresponding language. Average number of issues per perturbation text found by *language-tool-python*

bation paradigms including character level perturbation - DeepWordBug (Gao et al., 2018) and TextBugger (Li et al., 2019), BERT based word level perturbation - BAE (Garg and Ramakrishnan, 2020) and synonym swaps - TextFooler (Jin et al., 2020)¹. Besides measuring the number of successful and failed attempts, we also report the number of skipped attempts. An attempt is only counted as successful or failed if the classification of the original input is the same as the label, i.e $f(x) = y$. Thus, when $f(x) \neq y$, an attempt is labeled as *skipped*. The attempt success rate is then measured with:

$$success_rate = \frac{success_count}{success_count + failure_count} \quad (1)$$

Furthermore, as discussed in 3.2, we also use the paraphrase detector as an extra constraint for all of the successful perturbations. By using the paraphrase detector constraint, the successful perturbation not only fools the classifier but is also ensured to be paraphrase in of the original input taking all domain-specific (device functionalities) into account. Lastly, we calculate the perplexity as well as grammatical and semantic issues for perturbation texts from each method to see which method gives the best semantically sound texts. In terms of perplexity, we use *GPT-2* (Radford et al., 2019) for the SNIPS dataset, *mGPT* (Shliazhko et al., 2022) for MASSIVE dataset, *German GPT-2* (Schweter, 2020) and *French GPT2* for the German and French commercial dataset accordingly. All the GPT2 models are available at [Huggingface](https://huggingface.co). For grammatical issues, we calculate this metric by using

¹We take the implementation from TextAttack framework (Morris et al., 2020b) for our baselines

language-tool-python which is a wrapper of Language Tool, a grammar checker that counts the number of issues for every perturbation text and then gets the average number of issues. The results are shown in table 1.

For the effectiveness of the adversarial training we compare our adversarial data augmentation as described in 5.2 with a baseline model not trained on adversarial data and on a model trained using FGSM based adversarial training as described in (Shafahi et al., 2019), where we used the gradients to modify the input on the embedding layer to overcome the restrictions of the discrete nature of language input.

5 Results and discussion

In this section, we present our results on adversarial sample generation and adversarial training on top of the generated samples for the target models.

5.1 Adversarial sample generation

• **Attempt success rates.** We compare the four common text perturbation methods with our semi-supervised adversarial text perturbation method (SSAT) using two metrics: general success rate and success rate under the paraphrase detector constraint. The result of the experiment is shown in table 1. From the experimental results, SSAT’s success rate is comparable with other approaches as approximately 50% of the perturbation successfully fool the target model. However, when all perturbation recipes are subject to the paraphrase constraint, the SSAT method outperforms the rest of the recipes.

• **Attempt sample perplexity and grammar is-**

sues. For the perplexity metric, SSAT also consistently outperforms other baselines as shown in table 1, which indicates that the generated samples are more relevant and natural. In terms of fluency/grammatical issues of text samples, SSAT generated samples containing fewer grammatical issues than all 5 methods in comparison. This can be explained by our recipe using training data from the live-unannotated traffic data which is inherently more organic than using a masked language model like (Li et al., 2020) or rule based character swaps like (Li et al., 2019) without any domain adaptation. Indeed, when calculating the pairwise similarity between texts from each perturbation method and the unlabeled corpus dataset using *tf-idf*, SSAT’s generated samples are the most similar in cosine similarity to the unlabeled set of the dataset.

perturbation Method	tf-idf cosine similarity
BAE	0.171
DeepWordBug	0.155
TextBugger	0.152
TextFooler	0.057
SSAT	0.523

Table 2: pairwise similarity between all successful attempts from each perturbation method and the unlabeled set of MASSIVE dataset

To support our numbers with concrete examples, we picked some successful attempts across baselines and our method on the SNIPS dataset to highlight the difference in perturbation fluency. For example, synonym swaps methods like TextFooler can successfully perturbate a text by swapping in synonyms of some particular words. However, in practice, synonyms have to be taken in the context of the sentence for the swaps to be fluent and natural. While (*rate - rhythm*) and (*stars - celebs*) are synonyms, the swaps make the sentence unnatural and incomprehensible by a human reader as *rate* in this text is in the verb form with meaning of *evaluating*, not in the noun form as a musical term. On the other hand, by integrating unlabeled, human generated data into the adversarial generation process, SSAT generates a more natural variants like *give this textbook a three* given the sentence *rate this textbook a zero*. While the candidate does slightly change the complete meaning of the sentence, to a human reader, it keeps the overall meaning and the label classification of the text as both should be labeled as **rate book**. Other successful attempts from baselines such as DeepWordBug only made small modifications but the change completely alters the word’s meaning and make it hard to comprehend

to human readers. All the examples are shown in table 3.

Method	Input text	Perturbed text
BAE	i rate secret water as a 4	i use tap music as a 4
DeepWordBug	rate this current novel 1 stars	arte this current novel 1 stZars
TextBugger	rate maps for lost lovers 1 of 6	rate maps for lost lovers 1 of 6
TextFooler	rate lamy of santa fe 5 of 6 stars	rhythm lamy of santa fe 5 of 6 celebs
SSAT	rate this textbook a zero	give this textbook a three

Table 3: Some examples of successful perturbations from different baselines and our SSAT method on input texts from SNIPS dataset

Furthermore, we compare the change in the number of paraphrased pairs at different confidence levels of the paraphrase detector output. Figure 3

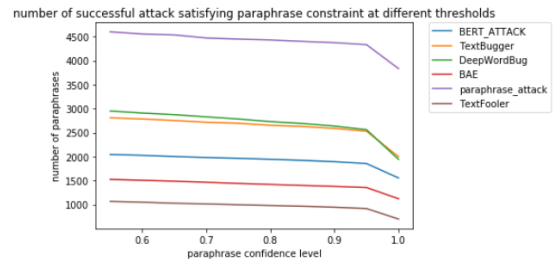


Figure 3: Number of successful attempt at different paraphrase detector threshold on the German commercial dataset. The experiment is run on 10, 000 perturbations.

showed that when gradually increasing the confidence threshold for paraphrase detector to judge that a pair of *input-perturbed text* are paraphrases, the number of successful attempts also gradually decreases. This is expected behavior as when the oracle model is more strict, the number of successful attempts should be decreasing. Another observation from the figure is that regardless of the threshold, UAT consistently performs better than all other perturbation recipes on the German commercial dataset.

• **Gradient-based targeted adversarial generation.** Similar to white-box adversarial perturbation methods such as HotFlip (Ebrahimi et al., 2018), we further experimented using token importance scores based on Integrated Gradients (IG) (Sundararajan et al., 2017) in the decoding step to check whether this could provide guidance on the generation step to find samples with higher attempt success rates. Specifically, we use the token importance score based on IG, which is calculated with respect to the predicted class of the original utterance to re-rank the probability of the next generated token in the beam search of the seq2seq

model. The IG value is positive if the token positively contributed to the predicted class, negative if the token is not associated with the predicted class and 0 if it is neutral. The idea is to generate and favor tokens that are more likely to switch classes to fool the target model. In our experiments, this however leads to poorer results, where we obtained attempt success rate of 51.1%, success paraphrase rate of 22.8%, perplexity of 618.9, and average grammatical issues of 2.22 on the German commercial dataset. This somehow shows that the decoding strategies of the translator model are sensitive to this reranking method, which leads to more invalid attempt generation, especially when running on discrete text input data, where a small local change in embedding gradient cannot account for a word replacement. We plan to experiment with other combinations of SSAT and white-box perturbation methods in the future to further understand this behavior.

5.2 Adversarial training

After generating adversarial instances, we include them to a vanilla adversarial training on the private dataset as a defense mechanism to improve the model’s robustness. For the German Commercial dataset, we ran adversarial perturbation on both the train and the test set with the target model being trained on the regular training set. The adversarial examples from training set are then combined with the original training set to train the target model. The so trained model is then tested on the original test set as well as the adversarial perturbations on test set and is compared with the original target model. In addition to the original target model, we also tested the effectiveness of a model trained with FGSM based adversarial training (Shafahi et al., 2019). From results in Table 4, the adversarially trained model only sacrificed a very small loss in standard test set performance but achieve a huge gain in the adversarial test set as the original target model is easily fooled by our method with performance of 15%. In comparison to that, the FGSM based adversarial training achieved slightly better performance than the vanilla model on the adversarial testset, however given the local perturbations used in this approach, this model is still fooled in 75% of the adversarial examples.

6 Conclusions

In this paper, we proposed a framework that uses pseudo-labeled data for learning and training an

Target Model	Class	Std test set	Adv test set
bert-base-german-cased FGSM-adv. trained Commercial dataset	Music	0%	19%
	Books	-1%	-3%
	Calendar	-1%	6%
	Shopping	-1%	15%
	Notification	7%	-1%
	Overall	-1%	10%
bert-base-german-cased SSAT adversarially trained Commercial dataset	Music	-0%	74%
	Books	-1.11%	61%
	Calendar	-2.13%	63%
	Shopping	-1.04%	73%
	Notification	7.6%	80%
	Overall	-1.03%	70%
bert-base-multilingual-cased FGSM-adv. trained MASSIVE dataset	Overall	0%	0%
bert-base-multilingual-cased SSAT adversarially trained MASSIVE dataset	Overall	-1.01%	25.23%

Table 4: The relative gain and loss in the accuracy performance of the adversarially trained model compared to the standard model on the standard test set and adversarial test set. The adversarial trained model sacrifices little in accuracy compared to standard model on regular test set and achieves large gain on adversarial tests

adversarial perturbation generator that can produce more relevant and natural samples. We trained a paraphraser detector to serve as an additional constraint to validate the generated perturbations. We showed that our perturbation methods outperforms general adversarial perturbation methods on success attempt rates and is able to generate meaningful samples with lower perplexity and less grammatical issues. We further demonstrated how applying adversarial training on the generated samples can better improve the model’s robustness than when using traditional gradient adversarial training such as FGSM.

7 Ethical Considerations

Our work proposes a new way of improving classification model performance in natural language understanding tasks. Since our approach is based on the usage of unlabeled data as it is occurring during production, there is a certain risk for the models to overfit on user groups that use the model the most which might introduce a bias for this group. In addition to that there is the need to keep the generation model of the adversarial perturbation generator current, making sure, that data that was removed by customers is also not used in any future application of the model.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang.

2018. [Generating natural language adversarial examples](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.
- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. [Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces](#). *CoRR*, abs/1805.10190.
- Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. [HotFlip: White-box adversarial examples for text classification](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.
- Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Nataraajan. 2022. [Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages](#).
- Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. [Black-box generation of adversarial text sequences to evade deep learning classifiers](#). In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.
- Siddhant Garg and Goutham Ramakrishnan. 2020. [BAE: BERT-based adversarial examples for text classification](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.
- Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. [Slot-gated modeling for joint slot filling and intent prediction](#). 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 753–757, New Orleans, Louisiana. Association for Computational Linguistics.
- Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. [Explaining and harnessing adversarial examples](#). In *International Conference on Learning Representations*.
- Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. [Adversarial example generation with syntactically controlled paraphrase networks](#). 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 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.
- Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. [Is bert really robust? a strong baseline for natural language attack on text classification and entailment](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. [Billion-scale similarity search with gpus](#). *IEEE Transactions on Big Data*, 7(3):535–547.
- Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. [Textbugger: Generating adversarial text against real-world applications](#). *Proceedings 2019 Network and Distributed System Security Symposium*.
- Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. [BERT-ATTACK: Adversarial attack against BERT using BERT](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.
- John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. [Reevaluating adversarial examples in natural language](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics.
- John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. [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.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language Models are Unsupervised Multitask Learners](#).
- 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.
- 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.
- Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. [Generating natural language adversarial examples through probability weighted word saliency](#). In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. [Semantically equivalent adversarial rules for debugging NLP models](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Stefan Schweter. 2020. [German gpt-2 model](#).

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32.

Oleh Shliakhko, Alena Fenogenova, Maria Tikhonova, Vladislav Mikhailov, Anastasia Kozlova, and Tatiana Shavrina. 2022. [mgpt: Few-shot learners go multilingual](#).

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mT5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. [Word-level textual adversarial attacking as combinatorial optimization](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. [PAWS: Paraphrase adversaries from word scrambling](#). 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 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

A FAISS search application

To apply FAISS search to our pseudo-labeled data, we first partitioned the pseudo-labeled data by the labels. Then, for each utterance x_i , we run FAISS *knn* search in one-vs-rest style where we search for the most similar utterances among the those

that do not share the same class label as x_i . Each of these similar sentences then got matched with x_i to generate k similar pairs for x_i

```

Input :  $X = [x_i], Y = [f(x_i)]$ 
SIMILARITYSEARCH( $X$ ):
   $S = \{\}$ 
  for  $x_i \in X$  do
     $S[x_i] = []$ 
    for  $y \in Y, y \neq f(x_i)$  do
       $classY = [x_j]$  s.t  $f(x_j) = y$ 
       $knn = \text{FAISS}(x_i, classY, k)$ 
      add  $knn$  to  $S[x_i]$ 
    end
  end
  return  $S$ 

CANDIDATE_PAIRS( $S$ ):
   $pairs = []$ 
  for  $x_i \in S$  do
     $nearestNeighbors = S[x_i] = [x_j]$ 
    add  $[x_i, x_j] \forall x_j \in S[x_i]$  to  $pairs$ 
  end
  return  $pairs$ 

```

Algorithm 2: Filtering live traffic with FAISS search

B Limitations

The limitation of our approach lies 1) in the post-filtering approach and 2) in the similarity to actually seen traffic. It is crucial and at the same time very hard to filter out those adversarial samples that are relevant and correct since the T5 model also produces sentences that do not preserve the meaning of the input or are grammatically correct. The second limitation lies in how similar the generated variations actually are to real variations actual customers would say. We are planning to investigate this further and test the target model trained on generated perturbed utterances on general data.

Is It Out Yet? Automatic Future Product Releases Extraction from Web Data

Gilad Fuchs*
eBay Research / Israel
gfuchs@ebay.com

Ido Ben-Shaul*
eBay Research / Israel
ibenshaul@ebay.com

Matan Mandelbrod
eBay Research / Israel
mmandelbrod@ebay.com

Abstract

Identifying the release of new products and their predicted demand in advance is highly valuable for E-Commerce marketplaces and retailers. The information of an upcoming product release is used for inventory management, marketing campaigns and pre-order suggestions. Often, the announcement of an upcoming product release is widely available in multiple web pages such as blogs, chats or news articles. However, to the best of our knowledge, an automatic system to extract future product releases from web data has not been presented. In this work we describe an ML-powered multi-stage pipeline to automatically identify future product releases and rank their predicted demand from unstructured pages across the whole web. Our pipeline includes a novel Longformer-based model which uses a global attention mechanism guided by pre-calculated Named Entity Recognition predictions related to product releases. The model training data is based on a new corpus of 30K web pages manually annotated to identify future product releases. We made the dataset openly available at <https://doi.org/10.5281/zenodo.6894770>.

1 Introduction

E-commerce marketplaces and online retailers are constantly updating their inventory with new products. Given the ever growing number of newly released products and their variety, it is becoming increasingly challenging to keep track of upcoming releases. Further, estimating which products are likely to become trendy and highly demanded is an additional task that becomes more difficult with the growth of online E-commerce. For E-commerce marketplaces, whose inventories often include an extremely large variation of products across thousands of different categories, the task of constantly tracking new product releases becomes presumably unfeasible without the leverage of automatic and

scalable solutions. In this paper, we demonstrate how an automatic ML-powered extracting pipeline can identify future product releases in billions of websites consisting of unstructured text and rank their demand with high accuracy. We define a future product release identification as identifying both the product name and either its exact release date or a time range.

Our pipeline includes the following main steps. First, the Common Crawl¹ monthly snapshot data is cleaned to include only text by using the pipeline describe in (Raffel et al., 2020; Xue et al., 2021) code². Specifically, we used the already cleaned dataset - “Colossal Clean Crawled Corpus” (C4) (Raffel et al., 2020) and the multilingual variant of the C4 dataset called mC4 (Xue et al., 2021). The next step is a simple but effective combination of data filtering with manually curated release-related key phrases (e.g. “will be released”). Next, a Named Entity Extraction (NER) model is used to detect possible product names and the corresponding releases dates. This step is followed by an additional filtering of non-product related releases using a novel Longformer-based model (Beltagy et al., 2020) (“text2release”) which classifies whether a web text indeed includes a future product release or not. We show that using the NER predictions to decide which tokens should have global attention improves the text2release model performance. Next, a consolidation phase aggregates the evidence collected from multiple websites to rank the most likely release date. Last, a buzz calculation for each product is performed based on counting the times each product appears in different websites. An overview of the entire pipeline can be seen in Figure 1. Experimental analysis shows that our pipeline can identify future product release date, in the range of 30 days, with an accuracy between

*These authors contributed equally to this work

¹<http://commoncrawl.org/>

²<https://github.com/google-research/text-to-text-transfer-transformerdataset-preparation>

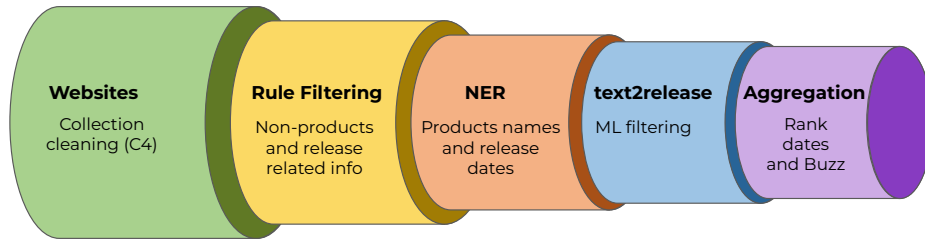


Figure 1: An overview of the product releases identification pipeline.

~70% to ~80%. In addition, our simple buzz calculation shows very high correlation with the actual product demand.

2 Related Work

2.1 Event Detection

There have been several works for predicting events from web data (Zhao, 2020). Some of these focus on discovering local and personal based events for individuals (Foley et al., 2015; Konovalov et al., 2017; Metzler et al., 2012; Li et al., 2017). In (Gravano and Becker, 2011)(Chapter 4), a method for identification of unknown events in social media sites based on trending occurrences is shown. This is done using incremental clustering algorithms, for finding event neighborhoods. Our proposed method is similar in theme to the work done in (Wang et al., 2019), where the aim is to build a database of global events. Other works have proposed to predict global events, mainly through use of data collected from social media platforms (Sakaki et al., 2010; Watanabe et al., 2011; Kim et al., 2018; Farzindar and Khreich, 2015).

In the E-commerce domain, (Yuan and Zhang, 2018) introduce a term frequency-inverse document frequency weighted word embedding to find relevant merchandises for seasonal retail events. However, they rely on a preset marketplace inventory. Finally, (Petrovski et al., 2014) proposes learning regular expressions for attribute extraction of E-commerce Microdata.

2.2 Classifying Long Sequences

The use of Transformers (Vaswani et al., 2017) in NLP applications has become extremely widely used, and accordingly in sequence classification. In general, transformer approaches are often limited to relatively short sequence size. Recently, the Longformer (Beltagy et al., 2020) was introduced to allow using the transformer mechanism on large documents, such as web-pages. Following this

work, the Big Bird (Zaheer et al., 2020) model was also proposed to handle the self-attention mechanism on long sequences. In both papers, a combination of the self-attention modifications is shown. In this paper we propose an additional method based on the Global Attention, where the tokens that receive global attention are based on outputs of an NER model. Other works which aim to deal with long sequence sizes have also been presented (Ainslie et al., 2020; Wang et al., 2020; Kitaev et al., 2020).

3 Future Product Releases Identification

3.1 Datasets

The C4 dataset³ described in (Raffel et al., 2020) was used for the entire product releases identification pipeline development and the ‘text2release’ model training. The C4 dataset is based on Common Crawl’s web data which was released in April 2019. The multi-lingual mC4 dataset (Xue et al., 2021) has 101 languages and is generated from 71 Common Crawl dumps. The product releases identification pipeline was tested over the newest snapshot from August 2020 and only English pages were selected (will be referred from now on as “Aug2020-Eng-mC4”).

3.2 Data Pre-Processing

As each monthly snapshot of the Common Crawl data may include hundreds of millions of web pages, we have decided to use a simple heuristics to select web pages which discuss future product releases. Our approach is based on manually curating a relatively broad set of key phrases which are likely to appear in future product releases web pages. The phrases chosen were general and potentially identify various types of future releases, not necessarily of products. Later stages in the pipeline further enrich the dataset by focusing specifically

³<https://www.tensorflow.org/datasets/catalog/c4>

on product releases. The key phrases and the corresponding number of pages including the specific phrase in the C4 dataset are listed in Appendix A.1, Table 6.

Although the C4 dataset is based on a snapshot from April 2019 of the Common Crawl corpus, the snapshot contains multiple pages from previous years which include future product releases that had already taken place. As our end goal is to detect on a monthly basis future product releases from the latest snapshot released by Common Crawl, we focused our methodology in identifying only future product release that occur after each snapshot release date. Specifically, we added a simple filter, on top of the previously described key phrases, requiring that the text explicitly includes a year string. The C4 dataset has been filtered to a subset containing the string “2019”. The Aug2020-Eng-mC4 dataset has been filtered for both the “2020” and “2021”, to identify products which were expected to be released at the end of 2020 or the at start of 2021. Although the year filtering may remove potentially relevant web pages that do not explicitly mention the year of release, it makes the dataset more relevant for the specific use-case aimed to be addressed by the pipeline. Following the key phrases and year filtering, the C4 and the Aug2020-Eng-mC4 datasets consist of ~292K and ~305K web pages, respectively.

Next, a subset of web pages were excluded based on a manually curated list of exclusion phrases which were identified to be dominant within releases-related texts and do not have applicable usage for our use case (e.g. mobile applications are usually not sold in E-commerce marketplaces). The main themes of the exclusion list phrases are related to mobile applications, music, TV, films and cars. Last, as manual probing of very long web pages revealed that those web pages rarely discuss future product release, and to ease the pipeline downstream processes, only web pages with text size shorter than 5000 characters were kept, which resulted in keeping ~75% of the web pages. Overall, following the pre-processing steps ~74K and ~78K web pages were selected from the C4 and Aug2020-Eng-mC4 datasets, respectively.

3.3 Entity Recognition of Products and Dates

In order to identify future product release it is essential to detect both the product name and its release date. While for some of the products the new

release might consist of only a new model of an existing product, often new releases are for entirely new products. For identifying a release of a new model for an existing product some heuristics can be used (e.g. looking for a pattern of a known product name + variation of a model number). For a previously unseen product such methods are not relevant. Hence, an NER model, capable of identifying product names based on the text context, was used. More specifically, we used the document level NER model FLERT (Schweter and Akbik, 2020), available as part of the Flair package (Akbik et al., 2019). FLERT was trained on the OntoNotes dataset (Weischedel et al., 2013), which includes 18 entity classes, and leverages document-level features by passing a sentence with its surrounding context. In our work we used the *PRODUCT*, *WORK OF ART* and *DATE* entities, as they were identified to be potentially relevant for our use case. Notably, the *WORK OF ART* entity was found to excel in identifying new books and video games specifically. The entity *DATE* was used to identify the different variations of dates described in web pages as free text. Only web pages where the FLERT model predicted the existence of either a *PRODUCT* or *WORK OF ART* were selected for the next step in the future product release identification pipeline. This additional filtering results in exclusion of approximately 50% and 43% of web pages in the C4 and Aug2020-Eng-mC4 datasets, respectively.

3.4 Future Product Releases Classifier

While the FLERT NER model predictions capture which pages include product entities, it can not assure that indeed the web page contains a description of a future product release. In addition, there are multiple cases where tokens are mistakenly predicted to be a *PRODUCT* or *WORK OF ART* entities. To further improve the identification of the web pages specifically describing future product releases, we created a new annotated dataset which includes approximately 30K web pages tagged by crowd-sourced labelers⁴. The 30K web pages were randomly sampled from the releases-enriched C4 dataset (following the steps described at Sections 3.2 and 3.3). Each page was labeled by 4 to 6 annotators, and the labelers were asked to select “text includes future product release” (~63% of pages) or “text doesn’t include a future product release”

⁴<https://doi.org/10.5281/zenodo.6894770>

(~37% of pages). Detailed description of the product releases dataset can be found in Appendix A.2.

In order to improve the identification of future product releases the annotated dataset was leveraged to train a classifier which detects texts mentioning a future product release (‘text2release’). As common modern text classification models (e.g. BERT (Devlin et al., 2018)) are limited to 512 sub tokens, and web pages are often significantly longer, we leveraged the pre-trained Longformer model which is capable of handling up to 4096 sub tokens. The tagged data was used to fine-tune the Longformer model, where only web pages having labeling confidence above 0.7 were used (~19,000 web pages). For validation and model testing, only pages with labeling confidence of 1 (i.e. all annotators agreed on the label) were used (~4,700 web pages).

While the original Longformer model uses for classification tasks a global attention in the first token only (specifically, the special ‘CLS’ token), we examined an alternative architecture which we coin “LongforNER” where global attention is assigned based on NER predicted entities. For this dataset, we chose *WORK OF ART* and *PRODUCT* entities from the FLERT model predictions. The assumption is that greater attention should be given to the product related text in order to better classify if a web page is about a future product release. In Table 1 we compare the test performance of the proposed model (LongforNER) with the results of the vanilla Longformer where global attention are assigned to the CLS token. The NER guided attention resulted in improved performance. It has been shown (Zaheer et al., 2020) that adding random global attention may assist during training to classify long texts. We therefore examined the impact of assigning randomly global attention to a subset of the tokens instead at the specific NER entities (see ‘Random’ in Table 1), to control the possibility that the improvement of the LongforNER performance is merely due to greater percentage of tokens with a global attention. We confirmed that the percentage of tokens which were assigned randomly with global attention was approximately the same as in the LongforNER version. The LongforNER version also showed better performance comparing to randomly assigned global attention. All models were trained for 30 epochs, with a batch size of 4 and a learning rate of $1 \cdot e^{-6}$, with cosine LR schedule and a minimum value of $5 \cdot e^{-8}$. An

Table 1: Comparing predicting future product releases performance metrics while assigning global attention in CLS (‘Longformer’), randomly (‘Random’) or based on NER predictions (‘LongforNER’). Each result is an average of 5 different random seed initialization. For the metrics that are based on a given threshold, we use the Youden Index (Youden, 1950). PR-AUC stands for Precision-Recall Area Under the Curve.

Metric	Longformer	Random	LongforNER
PR-AUC	0.8852	0.8834	0.8901
F1	0.7926	0.8059	0.8151
Accuracy	0.7284	0.7405	0.7481

AdamW (Loshchilov and Hutter, 2019) optimizer was used in all experiments. The text2release classifier predictions were used to further select web pages of higher probability to include future product release.

3.4.1 LongforNER Sequence Classification

To further test the advantage of the LongforNER architecture, we test it on the Hyperpartisan news detection dataset (Kiesel et al., 2019). Similar to (Beltagy et al., 2020), we focused on the ‘byarticle’ dataset, as it’s labels are of higher quality. The Hyperpartisan classification task is to decide whether a news article follows a hyperpartisan argumentation, i.e., whether it exhibits blind, prejudiced, or unreasoning allegiance to one party, cause, or person. The Hyperpartisan dataset was previously used to evaluate long texts classifiers (Beltagy et al., 2020; Zaheer et al., 2020). Intuitively, this dataset should benefit from global attention at named entities such as person or organization, as often news, and the hyperpartisan argumentation specifically, involves such entities (e.g. “President Trump and Republicans in Congress must act now to stop new Obamacare taxes...”). Hence, we use the flair⁵ 4 classes NER model which identifies the following entities: *PER* (person), *LOC* (location), *ORG* (organization), and *MISC* (other). In Table 2, we show the results of the LongforNER vs. the vanilla Longformer model, using the train/val/test given in (Beltagy et al., 2020). We found the split used in this work to be of particularly high performance. The authors show results on a single split using five different initializations, using the same train/val/test split. Hence, we also measured the performance following splitting the dataset with 5 random splits, using 5 different seed initializations for each. As done in

⁵<https://huggingface.co/flair/ner-english-large>

Table 2: Average Test F1 on 80/10/10 train/val/test split of the the HyperPartisan dataset using 5 different seed initialization. The split was done either as given in (Beltagy et al., 2020) (‘hyper-orig-split’) or 5 times randomly (‘hyper-new-split’).

Dataset	Longformer	Random	LongforNER
Hyper-orig-split	0.9350	0.9243	0.9390
Hyper-new-split	0.7638	0.7445	0.7822

Section 3.4, the LongforNER performance was also compared to a version where the global attention was assigned randomly. Overall, the LongforNER version shows better performance compared to the CLS-based global attention (Vanilla) and randomly assigned global attention (Random) for both types of the splits. All models were trained for 15 epochs, with a batch size of 4 and a learning rate of $2.5 \cdot e^{-5}$, with linear LR schedule. An AdamW (Loshchilov and Hutter, 2019) optimizer was used in all experiments.

3.5 Pipeline Consolidation

As one of the final goals of the pipeline is to identify the future product release date or a time range, it is necessary to convert the free text describing the date (identified by the NER model) to a structured date format. Specifically, we converted a single date point to a ‘DD/MM/YYYY’ format and a date range was converted to a tuple of (MIN(DATE), MAX(DATE)). A default of day=15 was used in cases where the release date includes only month and year without a specified day. While the simple patterns of free text dates were found to be parsed successfully with the open source package dateparser⁶, for more complicated patterns, which were found to be common in future releases texts, a custom parser was developed. The identified patterns used by the custom parser are summarized in Appendix A.3, Table 8.

As each web page might include several product names and dates, it is essential to link each product name to the corresponding release date. We employ a simple heuristic where we collect all pairs of identified *PRODUCT* or *WORK OF ART* with every *DATE* entity which appear in the same sentence. While this approach does not guarantee that the identified date is indeed the correct release, manual evaluation of sample candidates showed that this is often the case. Moreover, as pairs are collected

⁶<https://github.com/scrapinghub/dateparser>

Table 3: Example of 10 identified products (‘Product Name’), the suggested release date (‘Suggested Date’), and the number of supporting data points for the suggested date (‘Date Count’).

Product Name	Suggested Date	Date count
assassins creed valhalla	17/11/2020	164
far cry 6	18/02/2021	101
flight simulator	18/08/2020	81
cyberpunk 2077	19/11/2020	70
xbox series x	15/11/2020	66
wwe 2k battlegrounds	18/09/2020	66
kingdoms of amalur	08/09/2020	60
fifa 21	09/10/2020	58
nba 2k21	04/09/2020	51
watch dogs legion	29/10/2020	50

from multiple websites, aggregating the different dates per product reduces the noise by selecting the most frequent date per product. Any *PRODUCT* and *WORK OF ART* entities which did not have a *DATE* entity in the same sentence were filtered out.

Intuitively, the number of different websites discussing an upcoming product release should be at least partly correlated with the product demand upon its release. In order to count the number of web pages mentioning each product it is possible to count the mentioning of the specific identified product names across all the web pages of a Common Crawl snapshot. However, such a naive approach would yield a large number of false positives, as some product names are not specific enough. For example, searching for the video game “Control” in a full snapshot results in millions of websites. Therefore, we count only web pages where the NER model identified the text as a product name. We refer the number of web pages mentioning the product name as a ‘buzz’ calculation.

4 Experiments and Results

In order to test the product releases identification pipeline we used the Aug2020-Eng-mC4 dataset for evaluation, described in Section 3.1. Of note, the Aug2020-Eng-mC4 dataset was not used during any stage of the pipeline development, and mimics a case of fetching a new monthly snapshot from Common Crawl to identify future product releases. Running the product releases identification pipeline, as described in Section 1, on the Aug2020-Eng-mC4 dataset resulted in 243 overall products for which the release date was identified. Example of 10 identified products, can be seen in Table 3. Interestingly, 9 out of the top 10 identified prod-

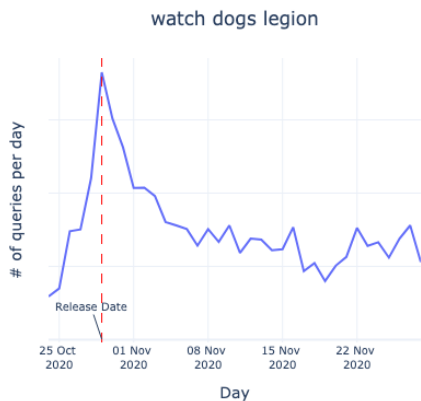


Figure 2: Search query logs per day for the video game ‘watch dogs legion’ during the 30 days following the game launch.

ucts are video games. We next manually annotated the categories of the 243 products. The number of identified products from each category are listed in Appendix A.3, Table 9. Overall, the pipeline results in enrichment of video games.

Next, the accuracy of the suggested release dates was evaluated. The true release date was manually labeled per product. Since most of the products were categorized as ‘Video Games’, ‘Smartphones’, ‘Electronics’ and ‘Books’, the manual labeling of the true release date was done only for the products belonging to these categories. We measure the percentage of products for which the suggested release date was identical to the true release date (‘P0’) and within the range of 10 or 30 days (‘P10’, ‘P30’). Of note, as in some cases only the expected month and year of the future product release date are mentioned in the text (e.g. “will be released in March, 2021”), this results in lower P0 and P10 compared to P30. Table 4 shows the accuracy of the suggested release dates for all products belonging to the top 4 categories (All) and for the largest category ‘Video Games’ specifically. The results show that an automatic ML-powered pipeline can identify the release date of more than 200 previously unknown products from a single month web-data snapshot with a error range of 30 days in approximately 70% accuracy. For the largest category of ‘Video Games’, which on average includes more supportive web pages per product release date compared to the rest of the categories (~10 vs ~4), the P30 accuracy is 78%.

Next, we examine if our buzz calculation can be used to predict at least partly future demand. In order to estimate the product demand we used eBay’s

Table 4: The % of products which their true release date was exactly as the identified release date (P0), within the range of 10 days (P10) or within the range of 30 days (P30) of the true release date.

	P0	P10	P30	N
All	52.7	63.2	69.5	220
Video Games	62.7	71.8	78.2	142

Table 5: Pearson and Spearman’s correlation between each product buzz calculation and the total number of search queries in 30 days since the product launch.

	Pearson	Spearman	N
All	0.873	0.647	83
Video games	0.923	0.788	56

search query logs. For each product, that had an actual release date during 2020 (but not before the Common Crawl snapshot release date of 16-Aug, 2020), the demand was estimated by the number of relevant queries found within 30 days since the release date. It is worth noting that not all products were found to be sold on eBay specifically. For simplicity, only products with a dominant single relevant query were examined. Figure 2 shows example of a release date correctly identified by the proposed method, and the query logs in the days after the release. Next, the correlation between the buzz calculation and the demand was calculated. As can be seen in Table 5, a high correlation was found between the buzz calculation and the actual demand, and even higher for products of the largest category of ‘Video Games’.

5 Conclusions

In this work we demonstrate the capability of automatically identifying future product releases and their ranked demand, from the free monthly snapshot of the Common Crawl data. The ability to identify product releases in advance is a powerful tool which can be leveraged for multiple downstream applications such as better management of inventory or price updates of outdated models. We also suggest a new NER-guided global attention mechanism to improve long text classification tasks. Last, we release a new dataset consisting of web pages labeled as whether the text includes future product releases or not.

References

- Joshua Ainslie, Santiago Ontañón, Chris Alberti, Václav Cvicek, Zachary Kenneth Fisher, Philip Pham, Anirudh Ravula, Sumit K. Sanghai, Qifan Wang, and Li Yang. 2020. Etc: Encoding long and structured inputs in transformers. In *EMNLP*.
- 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 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.
- Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. *Longformer: The long-document transformer*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. *Bert: Pre-training of deep bidirectional transformers for language understanding*.
- Atefeh Farzindar and Wael Khreich. 2015. A survey of techniques for event detection in twitter. *Computational Intelligence*, 31:132 – 164.
- John Foley, Michael Bendersky, and Vanja Josifovski. 2015. Learning to extract local events from the web. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Luis Gravano and Hila Becker. 2011. Identification and characterization of events in social media.
- Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, D. Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In **SEMEVAL*.
- Donghyeon Kim, Jinhyuk Lee, Donghee Choi, Jaehoon Choi, and Jaewoo Kang. 2018. Learning user preferences and understanding calendar contexts for event scheduling. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.
- Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451.
- Alexander Kononov, Benjamin Strauss, Alan Ritter, and Brendan T. O’Connor. 2017. Learning to extract events from knowledge base revisions. *Proceedings of the 26th International Conference on World Wide Web*.
- Cheng Li, Michael Bendersky, Vijay Garg, and Sujith Ravi. 2017. Related event discovery. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*.
- Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.
- Donald Metzler, Congxing Cai, and Eduard H. Hovy. 2012. Structured event retrieval over microblog archives. In *NAACL*.
- Petar Petrovski, Volha Bryl, and Christian Bizer. 2014. Learning regular expressions for the extraction of product attributes from e-commerce microdata. In *LD4IE@ISWC*.
- 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.
- Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *WWW ’10*.
- Stefan Schweter and Alan Akbik. 2020. *Fler: Document-level features for named entity recognition*.
- Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *ArXiv*, abs/1706.03762.
- Qifan Wang, Bhargav Kanagal, Vijay Garg, and D. Sivakumar. 2019. Constructing a comprehensive events database from the web. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*.
- Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *ArXiv*, abs/2006.04768.
- Kazufumi Watanabe, Masanao Ochi, Makoto Okabe, and Rikio Onai. 2011. Jasmine: a real-time local-event detection system based on geolocation information propagated to microblogs. In *CIKM ’11*.
- Ralph Weischedel, Martha Palmer, Mitchell Marcus, Eduard Hovy, Sameer Pradhan, Lance Ramshaw, Nianwen Xue, Ann Taylor, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Robert Belvin, and Ann Houston. 2013. *OntoNotes Release 5.0*.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. *mT5: A massively multilingual pre-trained text-to-text transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.
- W. J. Youden. 1950. Index for rating diagnostic tests. *Cancer*, 3.
- Ted Tao Yuan and Zezhong Zhang. 2018. Merchandise recommendation for retail events with word embedding weighted tf-idf and dynamic query expansion. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. *ArXiv*, abs/2007.14062.

Liang Zhao. 2020. Event prediction in big data era: A systematic survey. *ArXiv*, abs/2007.09815.

A Appendix

A.1 Data Pre-Processing

Table 6: Key phrases used to enrich releases-related web pages and the number of web pages consisting each key phrase.

Phrase	Number of web pages
"will be released"	556,702
"release date"	537,591
"to be released"	510,501
"will release"	321,214
"product launch"	199,099
"scheduled for release"	45,022
"will launch in"	36,411
"expected to launch"	33,985
"to come out in"	32,793
"release scheduled for"	1,795
Total	2,275,113

A.2 Product Releases Dataset

The product releases dataset is a new annotated dataset which includes approximately 30,000 web pages tagged by crowd-sourced labelers and openly available at <https://doi.org/10.5281/zenodo.6894770>. The dataset includes sampled web pages from the C4 dataset which were filtered as described in Sections 3.2 and 3.3. In this dataset however, only web pages with text size shorter than 3000 characters were kept (as opposed to 5000). The average number of characters per web page in the dataset is 1469 with a standard deviation of 721 and a median of 1412. The average number of tokens per web page is 244 with a standard deviation of 199 and a median of 236. The web page with the max number of tokens has 646 tokens. The average number of sub-tokens per web-page, using bert-base-uncased WordPiece tokenizer⁷, resulted in average of 324 sub-tokens with a standard deviation of 158 and a median of 312 sub-tokens. The web page with the max number of sub-tokens has 1869 sub-tokens. Of note, while the text2release model was trained on the product releases labeled dataset described in Section 3.4, it was also used

⁷<https://huggingface.co/bert-base-uncased>

to generate predictions on longer web pages (up to 5000 characters, as described in Section 3.2). Each page was labeled by 4 to 6 annotators with an average number of annotators of 4.7. The number of annotators per web page can be found in the column "judgments". The annotators were asked to tag each web page with "text includes future product release" or "text doesn't include a future product release". Approximately ~63% of the web pages were found to include a future product release. The annotators were guided to ignore releases that happened in the past, and future releases of non products entities of mobile applications, software, movies and TV shows.

Each web page is associated with a labeling confidence score. The confidence score was calculated by the Appen platform based on the level of agreement between multiple contributors (weighted by the contributors' trust scores). More details can be found in Appen website⁸. The average confidence score is 0.73 with a standard deviation of 0.146. The median confidence score is 0.743 and the number of web pages with confidence score of 1 is 4,688. Of note, as the annotators were required to label a relatively long and often complicated text, it is expected that not all annotators will agree on each of the label. Examples of positive and negative web pages can be seen in Table 7.

Table 7: Example of web pages labeled as "text includes future product release" (Positive) and "text doesn't include a future product release" (Negative).

Label	Text
Positive	"gladwell's previous five books (the tipping point, blink, outliers, what the dog saw, and david and goliath) have all been international bestsellers. in his ground-breaking blink, he explored the role of first impressions in our lives. now he goes deeper, zeroing in on how we make sense of the unfamiliar. talking to strangers will be published september 2019."
Negative	"get involved this spinal health week (20-26 may) to help raise awareness of the importance of being ready for life, so more australians can continue to do the things they love for longer. aca will release weekly blogs in the lead up to spinal health week ... tell us in 50 words or less how chiropractic helps you get ready for life, for your chance to win over \$700 worth of prizes including a garmin fitness tracker, bose wireless earbuds and a sunbeam stickmaster."

⁸<https://success.appen.com/hc/en-us/articles/201855939-How-to-Calculate-a-Confidence-Score>

A.3 Pipeline Experiments and Results

The patterns used by the custom date parser can be seen in Table 8. The number of identified products per category can be seen in Table 9.

Table 8: Patterns used to parse date ranges within free text.

Pattern	Example
the [first/second] half of YEAR	the first half of 2021
the [first/last] month of YEAR	the last month of 2019
the [beginning/end] of YEAR	the end of 2020
[early/late] YEAR	early 2021
the [first/.../last] quarter of YEAR	the forth quarter of 2020
[q1/q2/q3/q4] YEAR	q1 2021
MONTH [next/this] year	December this year
the [beginning/end] of MONTH	the end of August
[this/next] SEASON	this summer
the SEASON of [this/next] year	the winter of next year
the SEASON of YEAR	the fall of 2021

Table 9: The number of identified products per category.

Category	Number of Products
Video Games	142
Smartphones	29
Electronics	26
Books	23
Other	23

Automatic Scene-based Topic Channel Construction System for E-Commerce

Peng Lin*, Yanyan Zou*, Lingfei Wu, Mian Ma, Zhuoye Ding, Bo Long

JD.com, Beijing, China

{linpeng47,zouyanyan6,lingfei.wu,mamian,dingzhuoye,bo.long}@jd.com

Abstract

Scene marketing that well demonstrates user interests within a certain scenario has proved effective for offline shopping. To conduct scene marketing for e-commerce platforms, this work presents a novel product form, scene-based topic channel which typically consists of a list of diverse products belonging to the same usage scenario and a topic title that describes the scenario with marketing words. As manual construction of channels is time-consuming due to billions of products as well as dynamic and diverse customers' interests, it is necessary to leverage AI techniques to automatically construct channels for certain usage scenarios and even discover novel topics. To be specific, we first frame the channel construction task as a two-step problem, i.e., scene-based topic generation and product clustering, and propose an E-commerce Scene-based Topic Channel construction system (i.e., ESTC) to achieve automated production, consisting of scene-based topic generation model for the e-commerce domain, product clustering on the basis of topic similarity, as well as quality control based on automatic model filtering and human screening. Extensive offline experiments and online A/B test validates the effectiveness of such a novel product form as well as the proposed system. In addition, we also introduce the experience of deploying the proposed system on a real-world e-commerce recommendation platform.

1 Introduction

Recently, e-commerce platforms have become an indispensable part of people's daily life. Different from brick-and-mortar stores where salespersons can hold face-to-face conversations to promote products and even recommend more products related to customers' interests, most recommendation systems of e-commerce platforms, such as Taobao¹, mainly display individual products in

which users might be interested (Zhou et al., 2018, 2019), as listed in Figure 1 (indicated as Recommendation Flow Page). Recently, scene marketing has become a new marketing mode for product promotion where particular application scenarios (i.e., scene) are created to demonstrate product functions and highlight features correspondingly (Zhao, 2020), which is also paramount for e-commerce platforms to improve user experience during online shopping (Kang et al., 2019; Fu et al., 2019). A practical usage scenario of products can help users better understand product functions and features, and also allow the platform to exhibit more products that hit customer's specific interests, so that the user experience and click rate might be improved. However, scenes do not always help. For example, displaying all related products belonging to the same scene in the recommendation flow page might harm the user experience, since they tend to be homogeneous.

To achieve scene marketing in e-commerce platforms, this work presents a novel product form, scene-based topic channel, which consists of a list of diverse products belonging to the same scenario, together with two short phrases (or sentences) as the topic title summarizing the scene. Exemplified by Figure 1, one primary product of a channel and the associated scene topic title (highlighted with red box) are displayed in the recommendation flow page. If a user is interested in the primary product and clicks on it, the user is then redirected to the topic channel page where diverse products belonging to the same usage scenario are displayed. Existing ways to constructing scene-based topic channel mainly rely on expert knowledge and past experience of business operators in grouping products into different functional categories with certain scene topics (Mansell, 2002; Cooke and Leydesdorff, 2006; Fernandez-Lopez and Corcho, 2010). However, such methods are highly expensive with low efficiency and even impractical since there

*The first two authors made equal contributions. Correspond to Yanyan Zou.

¹<https://www.taobao.com/>



Figure 1: A screenshot of a scene-based topic channel on an e-commerce platform, with only four products due to limited space. Text with underline in the right-side Translation column are used to connect the translated words with associated parts in the topic channel.

are billions of products in the e-commerce platforms. Therefore, in this work, we propose an E-commerce Scene-based Topic Channel construction system (i.e., ESTC) to automatically construct such scene-based topic channels, where the task is framed as a two-step problem, i.e., scene-based topic generation and product clustering. One intuitive solution to obtaining scene topics is to make use of topic models (Blei et al., 2003; Roberts et al., 2013; Grootendorst, 2022) or techniques from extractive summarization (Basave et al., 2014; Wan and Wang, 2016), which are, however, restricted to assigning topics within a predefined limited candidate set, while there are often emerging scenes in the e-commerce fields. Thus, like Alokaili et al. (2020), we propose to generate scene-based topic titles for products, which allows to create novel topics not featured in the training set.

Nevertheless, in practice, the limitation of labeled data for training (around 5000 instances) hinders the generation quality of the model. On the other hand, we observe that generated topic titles, describing the same scenario, might be slightly different in formulation. Simply grouping products based on exact string match of generated topic titles results in channels with rare products. To address above issues, we first develop a pre-trained model in the e-commerce field to improve generation quality. Then, a semantic similarity based clustering method is designed to conduct product clustering to form the channel. Finally, to ensure the user

experience online, we further design a quality control module to strictly filter out undesired channels, such as inconsistent topic titles, or channels with irrelevant topic-product pairs. Our contributions are summarized as follows:

- A topic generation model in e-commerce field is proposed to generate scene-based topic titles for products, which is flexible to produce topics for emerging products and allows the system to discover novel scene topics.
- A semantic similarity based clustering method is designed to aggregate products with similar topic titles and form scene-based channels, which is able to improve the product diversity.
- A quality control module is designed to ensure the quality of the artificially constructed channels before they are released online.
- We introduce the overall architecture of deployed system where the ESTC has been successfully implemented into a real-world e-commerce platform.
- To the best of our knowledge, this is the first work on automatically constructing scene-based topic channel for scene marketing in e-commerce platforms.

2 Proposed Method

The development of the proposed ESTC system consists of three main parts, including scene-based topic generation for each products, scene-based

product clustering to aggregate products with similar topic titles, as well as the quality control module to ensure the quality of AI-generated channels. We also include a simple data augmentation module to discover weakly supervised data in order to improve the diversity of generated topic titles.

2.1 Scene-based Topic Generation

In this work, we propose to generate the scene-based topic titles for each product. To be specific, given input information $X = (x_1, x_2, \dots, x_{|X|})$ of a product P , including product’s title T , a set of attributes A and side information O obtained through optical character recognition techniques, paired with scene-based topic title $Y = (y_1, y_2, \dots, y_{|Y|})$, we aim to learn model parameters θ and estimate the conditional probability:

$$P(Y|X; \theta) = \prod_{t=1}^{|Y|} p(y_t|y_{<t}; X; \theta)$$

where $y_{<t}$ stands for all tokens in a scene title before position t (i.e., $y_{<t} = (y_1, y_2, \dots, y_{t-1})$).

Pretraining with E-commerce Corpus Pre-trained models (Radford et al., 2019; Devlin et al., 2019; Lewis et al., 2020; Raffel et al., 2020; Zou et al., 2020; Xue et al., 2021) have proved effective in many downstream tasks, however, most of which are developed on English corpora from general domains, such as news articles, books, stories and web text. In our scenario, we aim to produce topic titles in Chinese that summarize certain usage scenarios of products. Therefore, a model is required to understand the products through its associated information (such as title, semi-structured attributes) and generate scene-based topic titles, where we argue that the model should learn knowledge from e-commerce fields and thus propose to further pre-train models in domain (Gururangan et al., 2020). Specifically, besides the product title, attribute set as well as side information, we also collect the corresponding advertising copywriting of products from e-commerce platforms for the second phase of pre-training. We adopt the UniLM (Dong et al., 2019) with BERT initialization as backbone structure.

Recall that the product attributes A is a set without fixed order. We observe that input containing same attributes yet in different orders might results in different outputs. On the other hand, UniLM is an encoder-decoder shared architecture. To reinforce both the understanding and generation ability

of no-order input information, in addition to the original pre-training objectives of UniLM, we also propose two objectives to adapt the target domain:

- **Consistency Classification:** Given a product title-attributes pair, this task aims to classify if the two refer to the same product. For the positive example, the attributes and the title describe the same product and attributes are randomly concatenated as a sequence to introduce disorder noises. For the negative example, we randomly select attributes from a different product.
- **Sentence Reordering:** We split the product copywriting into pieces according to marks (such as comma and period). Such pieces are then shuffled and concatenated as a new text sequence. The model takes the shuffled sequence as input and learns to generate the original copywriting.

After the second phase of pre-training in the target e-commerce domain, we fine-tune the pre-trained model on the scene-based topic generation dataset.

2.2 Scene-based Product Clustering

One intuitive solution to constructing a scene-based topic channel is to group products with exactly the same generated topic titles. However, we observe there exists channels with similar topic titles, each of which merely contains several products, while we expect one channel has diverse products to ensure user experience. Therefore, we design a clustering module to aggregate products with semantically similar topic titles.

Topic Encoding To better learn scene-based topic representations and distinguish different topic titles, we take all topic titles from training set as input and employ the SimCSE (Gao et al., 2021) to further fine-tune the e-commerce pre-trained UniLM model in an unsupervised fashion. The embeddings of the last layer are used as the initialization for product clustering.

Product Clustering This module aims to group products with semantically similar topic titles into a cluster, in other words, a product list for a channel. Since we do not have prior knowledge of how many topic clusters the topic generation model would produce, we adopt the hierarchical clustering (Sahoo et al., 2006) where the number of kernels is

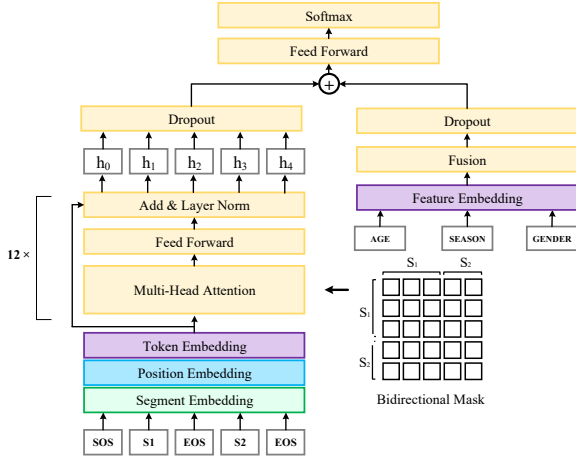


Figure 2: Correlation scoring model structure.

not required. To be specific, we adopt the bottom-up version, namely Agglomerative Nesting, which treats each sample as a leaf node and uses an iterative method for aggregation. In each iteration, two nodes with the highest similarity score are merged to form a new parent node. The iterative process stops when the shortest distance among all nodes is greater than a preset threshold. It is worthy noting each cluster might align with multiple topic titles and a list of products. The display order of products within a channel is decided by recommendation strategies, which is not focus of this work.

2.3 Quality Control

Although our method can generate good-quality channels most of the time, there is still possibility that the generated channels might not be accurate: 1) the generated topic title is semantically incoherent; 2) the topic title and associated products are not related according to the product usage scenarios. Thus, in order to alleviate above issues and ensure a reasonably good experience online, we design two modules, sentence coherence and correlation scoring models, to remove unexpected samples.

Topic Coherence Model We empirically observe that the generated topic titles might suffer coherent issues, like repetition and incompleteness. Thus, we design a topic coherence model to classify if a generated topic title is coherent. To be specific, the model is the e-commerce UniLM model with a softmax layer for classification. During training, we treat the online published topic titles as positive examples. The negative ones are synthesized:

- Samples with repetition: For a positive example of topic title, each unigram and bigram

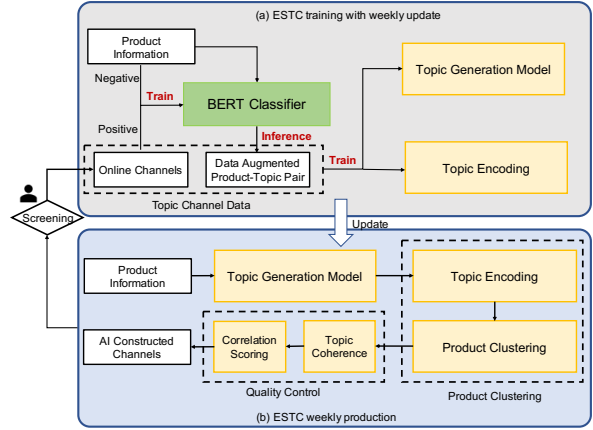


Figure 3: ESTC deployment on e-commerce platform.

is selected and repeated for one or two times with equal probability.

- Incomplete samples: We randomly remove the last two bigram or unigram tokens of a positive topic title.

We randomly select above synthesized samples to make the number of negative examples equal to the size of positive examples. Recall that a cluster might have multiple topic title candidates, the one with highest coherence score by the topic coherence model is used as final topic title. If all title candidates are classified as incoherent, then we simply remove such a cluster. After this module, each cluster is a scene-based topic channel with a list of products belonging to the same scene as well as a topic title summarizing the scene.

Correlation Scoring Model We design another binary classification model, i.e., correlation scoring model, to identify if the topic title and products are scene-based related. As illustrated in Figure 2, the e-commerce UniLM model takes as input the product information of a single product X as well as the generated topic title Y and determine whether they are related by the relevant scene. For better learning the product usage scenario, we also take into account the product profile information, such as age, season, and gender profiles, and employ a feed-forward layer to encode such features.

Likewise, product-topic title pairs from online published topic channels are considered as positive examples. The negative samples are obtained by randomly selecting mismatched product-topic title pairs. As a result, the number of negative examples is the same as the positive ones.

For each constructed channel, we use this model to check each product-topic title pair and remove

Model	SacreBLEU	ROUGE-1	ROUGE-2	ROUGE-L	BLEU	METEOR	DR(%)
BART	1.92	7.50	1.02	7.01	3.20	8.63	1.09
UniLM-BERT	2.05	7.87	1.11	7.42	3.45	8.70	0.87
E-commerce UniLM	2.08	8.01	1.12	7.56	3.47	8.78	0.88
E-commerce UniLM + DA	2.17	7.68	1.21	7.36	3.68	8.70	12.07

Table 1: The results of different topic generation model.

products that are unrelated to the topic.

2.4 Data Augmentation

Initially, the online existing (i.e., human-created) topic channels are quite limited which might hinder the model performance. Moreover, we would like to construct novel channels. Thus, we propose a UniLM-based binary classification model to discover more and diverse product-topic title candidate pairs. To be specific, the existing online product-topic title pairs are considered as positive examples. Similar to Zhang et al. (2022), a product with its the side information O from product detail images are considered as negative examples. After the classification model is trained, we use such a model to further extract more data for training. Negative examples with high probability scores are augmented into the training set.

3 Deployment

We have successfully deployed the proposed ESTC system on a real-world e-commerce platform. Figure 3 demonstrates the workflow of the deployed system with weekly update. Firstly, the data augmentation module is utilized to augment existing online channels. The augmented data is then used to train the topic generation and encoding models. Since there are thousands of millions products online, we weekly update the model and re-construct the channels to discover novel scene-based topic channels. To ensure a proper user experience, human screening is necessary before publishing channels online.

4 Experiment

4.1 Topic Generation Results

Data Collection The data for developing scene-based topic generation model consists of two sources: existing online channels (including human-created) and augmented samples, collected from a publicly available online e-commerce platform, JD.com². For the scene-based topic channels, we collected online channels from the product

²<https://www.jd.com/>

Dataset	#PT	#T	IL	OL
Human	177,412	5,186	69.34	13.44
Mined	111,572	82,834	74.21	12.54

Table 2: The statistics of topic generation dataset. #PT denotes the number of product-topic pairs, #T denotes the number of topic titles, IL denotes the average length of input product information sequence and OL denotes the average length of topic titles.

form ‘‘Goods List’’ from the platform, which were reviewed by human.

We also leveraged the optical character recognition (OCR) and classification techniques to extract key information about the product from product detail images. Firstly, texts are extracted from the images. Then, the extracted texts are ranked in descending order of their importance and relevance using the classification model. Finally, highly ranked texts are selected and merged as the final OCR input of products for topic generation.

In the end, we have 5,186 topic titles created by human and 82,834 topic title candidates from product side information. We further split the whole dataset into training, validation and test set with a ratio of 80%:10%:10%. The online channels are considered as ground-truth. Details are listed in Table 2. Moreover, we constructed product-OCR text (i.e., side information) pairs for the data augmentation module.

Comparison We use SacreBLEU (Post, 2018), ROUGE (Lin, 2004), BLEU (Papineni et al., 2002), and METEOR (Lavie et al., 2004) to measure quality of outputs by different generation models. We also design a new metric, difference rate (i.e., DR), to measure the novelty of generated topics, which is the ratio of the number of novel topics (i.e., not appearing in the training set) and the total number of generated ones. We consider publicly available models pre-trained on Chinese corpus as baselines, including BART (Shao et al., 2021) and UniLM with BERT initialization. As listed in Table 1, our E-commerce UniLM model achieves best performance for most evaluation metrics. With augmented data (denoted as +DA), the performance

Model	Silhouette	Recall	Precision	F1
B.O.W	0.264	90.8	88.7	88.0
Word2Vec	0.220	88.3	86.5	85.3
BERT	0.200	75.7	74.9	73.7
+SimCSE	0.283	88.9	89.0	88.1
E-commerce UniLM	0.248	72.3	68.7	68.0
+SimCSE	0.283	96.4	96.0	95.7

Table 3: The performance of different topic encoding models for clustering.

of our model is further improved with more novel topic titles produced, which shows the effectiveness of the data augmentation module.

4.2 Product Clustering Results

Dataset The clustering module works in an unsupervised fashion, while labeled data is still required for model evaluation. We manually create a data set for clustering evaluation, containing 65 different topic title samples, belonging to 18 groups.

Metrics We adopt the distance-based Silhouette Coefficient (Rousseeuw, 1987) to evaluate the performance of topic clustering. To investigate how well a clustering matches reference partitions of the test data, we further design two metrics.

For each topic sample i from cluster j , the precision score is calculated as:

$$P_{i,j} = \frac{TP_{i,j}}{N_j} \quad (1)$$

where $TP_{i,j}$ denotes the number of correctly grouped topic i in cluster j , N_j is the number of samples in cluster j . Similarly, the recall score is calculated as:

$$R_{i,j} = \frac{TP_{i,j}}{T_i} \quad (2)$$

where T_i is the total number of topic i found across all clusters.

The F1-measure score is computed as the harmonic mean of precision and recall.

Comparison We compare different sentence embedding-based clustering methods, including bag of words (i.e., B.O.W), Word2Vec (Mikolov et al., 2013), BERT as well as our E-commerce UniLM model. As listed in Table 3, models with SimCSE achieve better clustering performance. It is worthy noting that, the Silhouette score is not consistent with our designed metric scores. We practically observed that higher F1 scores indicate better clustering results for topics.

4.3 Quality Control Results

We also conducted human evaluation to investigate the effectiveness of each module for quality control, where for each setting, 1000 constructed channels are presented for human screening and report overall acceptance rate that is the ratio of the validated channels and the all candidates. As listed in Table 5, considering both topic coherence and correlation scoring modules results in the highest acceptance rate, demonstrating strengths of quality control module.

4.4 Experiments of Different Clustering Methods

As listed in Table 4, we compare two clustering method, K-means and hierarchical clustering methods, where the initial embedding are taken from different models. The hierarchical clustering with SimCSE enhanced e-commerce UniLM model achieves best performance.

4.5 Online A/B Test

To demonstrate the payoff generated by ESTC system, a standard A/B testing is conducted to evaluate the benefit of deploying scene-based topic channels on an e-commerce mobile app. After launching such a new product form, the Click-Through Rate (CTR) is improved by 3.20%, compared to the one without scene-based topic channels, which shows the values of AI-generated scene-based topic channels. We note that the comparison between human-created and AI-generated channels is difficult to fairly determine, since there are many factors mattering the online performance, such as recommendation strategies of products within channels.

More details about generated samples are included in Appendix.

5 Lessons Learned During Deployment

Several lessons we have learned during model deployment could be beneficial for other like-minded practitioners who wish to deploy cutting-edge AI technologies into real-world applications, such as the importance of real-world data quality and business understandings.

- Data quality matters model performance. Besides the model capacity, the quality of training data is of paramount importance. The cleaning procedures of raw data (e.g., removing poor samples from training set and speci-

Model	Kmeans				HC			
	Silhouette	Recall	Precision	F1	Silhouette	Recall	Precision	F1
B.O.W	0.221	81.9	74.3	72.4	0.264	90.8	88.7	88.0
Word2Vec	0.191	82.3	80.3	78.7	0.220	88.3	86.5	85.3
BERT	0.150	68.3	64.8	62.7	0.200	75.7	74.9	73.7
+SimCSE	0.262	90.0	86.2	86.6	0.283	88.9	89.0	88.1
E-commerce UniLM	0.210	73.5	69.0	68.4	0.248	72.3	68.7	68.0
+SimCSE	0.248	84.5	83.1	81.9	0.283	96.4	96.0	95.7

Table 4: The performance of different topic encoding models and different clustering models.

fying important attributes) plays a critical role in model development.

- Business understandings and logics advance AI model launching. The AI constructed scene-based topic channels are not fool proof. Thus, in order to ensure a reasonably good user experience, post-processing, based on insightful business understandings and logics, of AI constructed channels in the production platform is necessary to filter out any inconsistent or low-quality contents.

6 Related Work

Previous studies (Lau et al., 2011; Bhatia et al., 2016; Mei et al., 2007) on topic mining mainly first retrieve candidate topic labels from reference corpora and then conduct topic ranking to select the best topic label. Lau et al. (2010) simply take a word from a top-N terms as the topic label. Knowledge bases are also adopted to retrieve topic labels by matching candidate topic words to knowledge concepts (Magatti et al., 2009; Hulpus et al., 2013). Techniques from extractive summarization have also been used for topic extraction (Basave et al., 2014; Wan and Wang, 2016), which typically extract summary sentences from the input text related to topics. Recent years have witnessed neural networks are successfully leveraged to improve performance of topic modeling techniques, such as incorporating neural embeddings into existing LDA-like models (Bianchi et al., 2021; Thompson and Mimno, 2020), as well as the clustering embedding based approaches (Sia et al., 2020; Angelov, 2020; Grootendorst, 2022). A potential limitation of such methods is that the topic labels are within a predefined limited candidate set, while there are often emerging scenes in the e-commerce fields. Therefore, similar to Alokaili et al. (2020), we design a pre-trained model in e-commerce domain to generate scene-based topic titles, which allows to generate novel topics not featured in training set.

Architecture	Acceptance Rate (%)
ESTC w/o Quality Control	51.6
+ Topic Coherence	65.6
+ Correlation Scoring	60.6
+ Both	75.0

Table 5: Human evaluation for quality control.

Natural language processing techniques have been widely used in e-commerce fields to improve user experience, including automatic product copywriting generation (Zhang et al., 2022; Wang et al., 2022), online product review generation (Fan et al., 2019; Liu et al., 2021) and question generation (Gao et al., 2020; Deng et al., 2020). Differently, we propose to leverage natural language generation and clustering techniques to automatically construct scene-based topic channels, which, to the best of our knowledge, is novel.

7 Conclusion

This work aims to automatically construct scene-based topic channels. According to the understanding of business requirements, we propose to first generate topic titles for each product following by conducting product clustering to form a channel and design a novel framework, consisting of topic generation, product clustering and post-processing modules. The extensive offline experiments and online A/B test have demonstrated the effectiveness of the proposed approach. Incorporating user behaviors (e.g., click preference) into channel construction processes is worthy investigating in future. For example, the generated title and the clustered product are personalized.

8 Ethical Considerations

The data used in this work are collected from a publicly available online e-commerce platform, where the collection process is consistent with the terms of use, intellectual property and privacy rights of the platform. The annotated data for clustering evaluation are constructed by authors, where the

process is fair for all models. Please note that no private user data was used during data collection process.

The proposed ESTC system can be deployed on various e-commerce platforms where the scene marketing is required. On the other hand, the display style (or the product form) can be changed according to the practical needs, where the proposed system can provide products that belong to same usage scenarios.

Moreover, the AI constructed channels are not fool proof. Thus, as we discussed in the paper, in order to ensure the users can have a reasonably good experience, quality control and human screening of AI generated channels in the production platform is necessary to filter out any inconsistent or low-quality content.

9 Acknowledgements

We thank all the anonymous reviewers for their constructive comments.

References

- Areej Alokaili, Nikolaos Aletras, and Mark Stevenson. 2020. [Automatic generation of topic labels](#). In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 1965–1968. ACM.
- Dimo Angelov. 2020. Top2vec: Distributed representations of topics. *arXiv preprint arXiv:2008.09470*.
- Amparo Elizabeth Cano Basave, Yulan He, and Ruifeng Xu. 2014. Automatic labelling of topic models learned from twitter by summarisation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 618–624.
- Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. 2016. Automatic labelling of topics with neural embeddings. In *International Conference on Computational Linguistics*. Association for Computational Linguistics, ACL Anthology.
- Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2021. Cross-lingual contextualized topic models with zero-shot learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1676–1683.
- David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. [Latent dirichlet allocation](#). *J. Mach. Learn. Res.*, 3:993–1022.
- Philip Cooke and Loet Leydesdorff. 2006. Regional development in the knowledge-based economy: The construction of advantage. *The Journal of Technology Transfer*, 31:5–15.
- Yang Deng, Wenxuan Zhang, and Wai Lam. 2020. Opinion-aware answer generation for review-driven question answering in e-commerce. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 255–264.
- 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.
- Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 13063–13075.
- Miao Fan, Chao Feng, Lin Guo, Mingming Sun, and Ping Li. 2019. Product-aware helpfulness prediction of online reviews. In *2019 World Wide Web Conference, WWW 2019*, pages 2715–2721. Association for Computing Machinery, Inc.
- Mariano Fernandez-Lopez and Oscar Corcho. 2010. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Publishing Company, Incorporated.
- Min Fu, Qiang Chen, Wei Lin, Pei Wang, and Wei Zhang. 2019. Constructing a scene-based knowledge system for e-commerce industries: Business analysis and challenges. *Data Intelligence*, 1(3):224–237.
- Shen Gao, Xiuying Chen, Chang Liu, Li Liu, Dongyan Zhao, and Rui Yan. 2020. Learning to respond with stickers: A framework of unifying multi-modality in multi-turn dialog. In *Proceedings of the Web Conference 2020*, pages 1138–1148.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [Simcse: Simple contrastive learning of sentence embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.
- Maarten Grootendorst. 2022. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*.

- Suchin Gururangan, Ana Marasovic, 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, ACL 2020, Online, July 5-10, 2020*, pages 8342–8360. Association for Computational Linguistics.
- Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using dbpedia. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 465–474.
- Wang-Cheng Kang, Eric Kim, Jure Leskovec, Charles Rosenberg, and Julian McAuley. 2019. Complete the look: Scene-based complementary product recommendation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.
- Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 1536–1545.
- Jey Han Lau, David Newman, Sarvnaz Karimi, and Timothy Baldwin. 2010. Best topic word selection for topic labelling. In *Coling 2010: Posters*, pages 605–613.
- Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. 2004. The significance of recall in automatic metrics for mt evaluation. In *AMTA*, pages 134–143.
- 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.
- Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.
- Junhao Liu, Zhen Hai, Min Yang, and Lidong Bing. 2021. Multi-perspective coherent reasoning for helpfulness prediction of multimodal reviews. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5927–5936.
- Davide Magatti, Silvia Calegari, Davide Ciucci, and Fabio Stella. 2009. Automatic labeling of topics. In *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications*, pages 1227–1232.
- Robin Mansell. 2002. Constructing the knowledge base for knowledge-driven development. *Journal of Knowledge Management*, 6(4):317–329.
- Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 490–499.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.
- Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.
- 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.
- Margaret Roberts, Brandon Stewart, Dustin Tingley, and Edoardo Airoldi. 2013. The structural topic model and applied social science. *Neural Information Processing Society*.
- Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65.
- Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. 2006. Incremental hierarchical clustering of text documents. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 357–366.
- Yunfan Shao, Zhichao Geng, Yitao Liu, Junqi Dai, Fei Yang, Li Zhe, Hujun Bao, and Xipeng Qiu. 2021. Cpt: A pre-trained unbalanced transformer for both chinese language understanding and generation. *arXiv preprint arXiv:2109.05729*.
- Suzanna Sia, Ayush Dalmia, and Sabrina J Mielke. 2020. Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too! In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1728–1736.
- Laure Thompson and David Mimno. 2020. Topic modeling with contextualized word representation clusters. *arXiv preprint arXiv:2010.12626*.

Xiaojun Wan and Tianming Wang. 2016. Automatic labeling of topic models using text summaries. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2297–2305.

Zeming Wang, Yanyan Zou, Yuejian Fang, Hongshen Chen, Mian Ma, Zhuoye Ding, and BO Long. 2022. Interactive latent knowledge selection for e-commerce product copywriting generation. In *Proceedings of The Fifth Workshop on e-Commerce and NLP (ECNLP 5)*, pages 8–19.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. **mT5: A massively multilingual pre-trained text-to-text transformer**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Xueying Zhang, Yanyan Zou, Hainan Zhang, Jing Zhou, Shiliang Diao, Jijia Chen, Zhuoye Ding, Zhen He, Xueqi He, Yun Xiao, et al. 2022. Automatic product copywriting for e-commerce. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Mingxiao Zhao. 2020. Data-driven scene marketing based on consumer insight. In *2020 International Conference on Big Data Economy and Information Management (BDEIM)*, pages 61–65. IEEE.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1059–1068.

Yanyan Zou, Xingxing Zhang, Wei Lu, Furu Wei, and Ming Zhou. 2020. Pre-training for abstractive document summarization by reinstating source text. In *Proceedings of the Empirical Methods in Natural Language Processing*.

A Experiment Settings

A.1 The Hyper-parameters of Scene-based Topic Generation Model

In this section, we introduce the detailed setting of proposed Scene-based Topic Generation. To generate the scene-based topic titles for each product, we design a topic generation model based on UniLM, the input sequence and output sequence are encoded by the same attention module with different attention masks. The model is a 12-layer

transformer with multi-head attentions. During training, the learning rate is 0.00007, the warmup proportion is set to 0.2 and the batch size is 1024. The detailed hyper-parameters are listed in Table 6. The rest of the parameters are set by default.

hyper-parameters	value
learning_rate	0.00007
warmup_proportion	0.2
batch_size	1024
max_input_length	120
max_output_length	20
beam_size	4
embedding_size	768
hidden_dropout_prob	0.1
hidden_size	768
layer_norm_eps	1e-12
max_position_embeddings	250
num_attention_heads	3
num_hidden_layers	12
activation	"gelu"
vocab_size	21128

Table 6: The detailed hyper-paramters of architecture of E-commerce UniLM.

A.2 The Hyper-parameters of Topic Encoding Model

The topic encoding model encodes topic texts into vector features of specified dimensions, which facilitates clustering by common numerical clustering models. In this section, we introduce the detailed setting of proposed theme encoding model. We first obtain 700k topic data by the inference results of the E-commerce UniLM model. Then we employ SimCSE³ to fine-tune the second pre-trained UniLM model in the e-commerce domain. The backbone model is a 12-layer transformer. The learning rate is 0.00003 and the batch size is 64. The rest of the parameters are set by default.

hyper-parameters	value
num_train_epochs	4
max_len	32
train_batch_size	64
learning_rate	3e-5
max_seq_length	32
evaluation_strategy	steps
pooler_type	cls
temp	0.05

Table 7: The detailed hyper-paramters of architecture of topic encoding.

³<https://github.com/princeton-nlp/SimCSE>

A.3 The Hyper-parameters of Topic Coherence Model

Topic Coherence Model is 12-layer transformer with a feed-forward network and a softmax layer to distinguish whether the input topic is coherent. The learning rate is 0.00005 and the batch size is 2048. The rest of the parameters are set by default.

hyper-parameters	value
learning_rate	0.00005
warmup_proportion	0.1
batch_size	2048
max_len	32
embedding_size	768
hidden_dropout_prob	0.1
hidden_size	768
layer_norm_eps	1e-12
max_position_embeddings	250
num_attention_heads	3
num_hidden_layers	12
activation	"gelu"
vocab_size	21128

Table 8: The detailed hyper-parameters of architecture of topic coherence model.

A.4 The Hyper-parameters of Correlation Scoring Model

To filter out bad cases where topic title is not suitable for product usage scenarios, we design another binary classification model, i.e., correlation scoring model, to identify if the topic title and products are scene-based related. We concatenate the product description information X and topic title Y as the input of UniLM. For better learning the product usage scenario, we also take into account the product profile information, such as age, season, and gender profiles, and employ an embedding layer and a feed-forward layer to encode such features. The detailed hyper-parameters of correlation scoring model are listed in Table 9. The rest of the parameters are set by default.

B The Generated Scene-based Topic

In this subsection, we show some example topics generated by topic generation models, as well as examples of topics generated by the entire system, as shown in Table 10 and Table 11, respectively.

C Examples of Scene Marketing

In recent years, many companies have begun to use scene marketing to promote products, as shown in

hyper-parameters	value
learning_rate	0.00005
warmup_proportion	0.1
batch_size	2048
max_len	158
feature_embedding_size	300
feature_fusion_size	300
feature_vocab_size	13
embedding_size	768
hidden_dropout_prob	0.1
hidden_size	768
layer_norm_eps	1e-12
max_position_embeddings	250
num_attention_heads	3
num_hidden_layers	12
activation	"gelu"
vocab_size	21128

Table 9: The detailed hyper-parameters of architecture of correlation scoring model.

Figure 4, which are scene marketing for IKEA⁴ and Amazon⁵. IKEA combines different types of furniture in a scene room to highlight key attributes of furniture, such as storage and simple shape. Amazon also exhibits the functional scenarios for products, like babysitting and party games. Such scene marketing can help consumers understand the functions and features of products, which may improve user experience and product conversion rates.

⁴<https://www.ikea.com/>

⁵<https://www.amazon.com/>


Input	Generated Topic
创意烟灰缸生日送礼送男朋友 Creative ashtray birthday gift for boyfriend	送男友好物 @ 为爱精挑细选 Gifts For Boyfriends @ Carefully Selected For Love
夏季宽松休闲翻领男上衣 Summer Loose Casual Lapel Men's Top	精选t恤 @ 夏日清凉出行 Selection Of T-shirts @ Summer Cool Outting
网红款渔夫帽 Web celebrity's fisherman hat	防晒合集 @ 清凉防晒一夏 Sun Protection Collection @ All Summer Cool Sun Protection
龙井2022新茶绿茶茶礼盒装 Longjing 2022 new tea green tea gift box	女婿必买 @ 教你一招搞定老丈人 Son-in-law Must Buy @ Teach You A Trick To Get Father-in-law

Table 10: The generated scene-based topic titles by the topic generation model. We use @ to separate two phrases of the topic title.


Generated Topic	Product List
初生好礼 @ 虎娃新生儿礼物 Newborn Gift @ Tiger Baby Newborn Gift	尿裤尿不湿学步裤吸湿透气 Moisture absorbent breathable diaper toddler pants 初生宝宝幼儿浴巾被子防惊跳睡袋 Newborn baby bath towel quilt anti-startle sleeping bag 婴儿记忆棉乳胶枕头枕芯 Baby memory foam latex pillow 婴儿配方奶粉2段850克 Infant formula milk powder 2 stage 850g
快乐露营 @ 露营运动欢乐时光 Happy Camping @ Camping Sports Happy Hour	露营灯强光手电筒帐篷灯 Camping lights glare flashlight tent lights 户外折叠桌椅便携式野外可折叠野餐桌子 Portable outdoor folding picnic table 登山露营保暖加宽双人户外棉睡袋 Mountaineering camping warm widening double outdoor cotton sleeping bag 大空间防风3-4人三秒速开全自动速搭帐篷 Large space windproof 3-4 people three seconds to open fully automatic tent
尽情挥洒汗水 @ 是兄弟一起上球场 Sports Sweat @ On The Court With Your Brother	高帮板鞋男子经典运动休闲鞋篮球文化鞋 High-top sneakers men's classic sneaker, basketball culture shoes 男装梭织运动长裤运动服男 Men's woven sports trousers sportswear 简约经典训练系列男子圆领套头休闲百搭卫衣 Simple and classic training series casual all-match sweatshirt 针织五分裤男透气舒适夏季短裤男运动裤子 Knitted 1/2 pants men's breathable and comfortable summer running workout joggers

Table 11: The generated scene-based topic channel of ESTC system. We use @ to separate two phrases of the topic title.









Examples of Scene Marketing for Different Companies




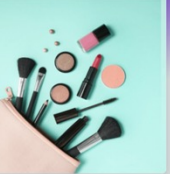


PAX bedroom storage furniture collection
A flexible closet collection that offers clothing solutions to fit different sized spaces. Keeping all your clothes neatly organized and in their place, while still showing off your individual style.



MALM series
Simple shape, versatile with various home styles.

Must have Baby products		Adult party games	
			
			

Shop Laptops & Tablets 	Oculus 	Women's Fashion 	Beauty Picks 
---	--	--	---






Figure 4: Examples of scene marketing for different companies.

SpeechNet: Weakly Supervised, End-to-End Speech Recognition at Industrial Scale

Raphael Tang,¹ Karun Kumar,¹ Gefei Yang,¹ Akshat Pandey,¹ Yajie Mao,¹
Vladislav Belyaev,¹ Madhuri Emmadi,¹ Craig Murray,¹ Ferhan Ture,¹ Jimmy Lin²

¹Comcast Applied AI ²University of Waterloo

¹firstname_lastname@comcast.com ²jimmylin@uwaterloo.ca

Abstract

End-to-end automatic speech recognition systems represent the state of the art, but they rely on thousands of hours of manually annotated speech for training, as well as heavyweight computation for inference. Of course, this impedes commercialization since most companies lack vast human and computational resources. In this paper, we explore training and deploying an ASR system in the label-scarce, compute-limited setting. To reduce human labor, we use a third-party ASR system as a weak supervision source, supplemented with labeling functions derived from implicit user feedback. To accelerate inference, we propose to route production-time queries across a pool of CUDA graphs of varying input lengths, the distribution of which best matches the traffic's. Compared to our third-party ASR, we achieve a relative improvement in word-error rate of 8% and a speedup of 600%. Our system, called SpeechNet, currently serves 12 million queries per day on our voice-enabled smart television. To our knowledge, this is the first time a large-scale, Wav2vec-based deployment has been described in the academic literature.

1 Introduction

Training an end-to-end automatic speech recognition (ASR) model requires hundreds, if not thousands, of hours of hand-labeled speech. With the rise of silicon-hungry pretrained transformers, these models additionally need increasing amounts of computational power just to perform inference. Together, these two hurdles impede effective model deployment at all but the largest technology companies and specialized speech processing startups. The hurdles certainly apply to us at Comcast, the main stage of this work. Our industrial challenge is to fine-tune and deploy a large, pretrained speech recognition model, without an army of annotators (as in Amazon) or mammoth GPU farms (e.g., Google). Our end application is the Xfinity X1,

a voice-enabled smart television serving millions of active devices in the United States.

Evidently, cloud ASR services are cheaply available.¹ Google Cloud, for example, charges \$1.44 USD per hour of transcribed speech. In contrast, manual annotation services like Rev cost \$90 *per hour*, and our in-house annotators, whom Comcast must use to protect user privacy, cost even more. Thus, cloud ASR's comparatively low pricing, combined with its decent quality, suggests its utility as an annotation source in the absence of substantial human-labeled data.

Nevertheless, cloud ASR still falls short of human parity and hence demands label denoising. To do this, we propose to use implicit user feedback to remove incorrectly labeled examples, bootstrapping an existing cloud ASR service. We derive these labeling functions using signals from query repetition, session length, and ASR confidence scores. We model them in Snorkel (Ratner et al., 2017), a popular data programming framework, producing a 1400-hour weakly labeled dataset. Trained on this, our models improve over those using unfiltered data by an average 0.97 points in word-error rate (WER), as presented in Section 4.

As for the second hurdle of resource efficiency, many model acceleration methods exist. However, few meet our productionization criteria: we seek to preserve the quality, ruling out structured pruning (Li et al., 2020); we wish to preserve the pretrained architectural structure, eliminating knowledge distillation (Tang et al., 2019a); and we require stable software-GPU support, disqualifying low bit-width quantization (Shen et al., 2020) and other CPU-oriented approaches.

All things considered, the prime candidates are medium bit-width quantization, decoder optimizations (Abdou and Scordilis, 2004), and CUDA computation graphs (Gray, 2019). The first two follow

¹But not cheaper or better than using our own in-house ASR system; otherwise, there would be no need for this work!

the literature, but the third is more open ended. In spite of their record-breaking performance, CUDA graphs work only with fixed-length input, not variable length. Toward this, we propose to allocate a pool of CUDA graphs of varying lengths, altogether matching the production-time traffic length distribution. During inference, we route each query to the graph with the least upper-bound in length. As we show in Section 4, this yields a 3–5× increase in throughput.

We claim the following contributions: first, we derive novel labeling functions for constructing weakly labeled speech datasets from in-production ASR systems, improving our best model by a relative 8% in word-error rate. Second, we propose to accelerate model inference using a pool of CUDA graphs, attaining a 7–9× inference speed increase at no quality loss. The resulting system, SpeechNet, currently serves more than 20 million queries per day on our smart television. To our knowledge, we are the first to describe a large-scale, Wav2vec-based deployment in the academic literature.

2 Our SpeechNet Approach

Our task is to train and deploy a state-of-the-art, end-to-end ASR system, without using human-annotated data. The context of this deployment is a smart TV, which users interact with using a speech-driven remote control. To issue a voice query, users hold a button, speak their command, and release the button. We initially serve them with a third-party cloud ASR service, bootstrapping it for the development of SpeechNet. Data-wise, we store thousands of hours of utterances per day, complete with session IDs, transcripts, and device IDs. Resource-wise, we have 30 deployment nodes, each hosting an Nvidia Tesla T4 GPU and receiving 120 queries per second (QPS) at peak time; thus, our model’s real-time factor must exceed 120.

2.1 End-to-End ASR Modeling

In end-to-end ASR systems, we transcribe speech waveform directly to orthography, consolidating the traditional acoustic–pronunciation–language modeling approach. Similar to natural language processing, the dominant paradigm in speech is to pretrain transformers (Vaswani et al., 2017) on unlabeled speech using an unsupervised contrastive objective, then fine-tune on labeled datasets (Baevski et al., 2020). We practitioners further fine-tune these released models on our in-domain datasets.

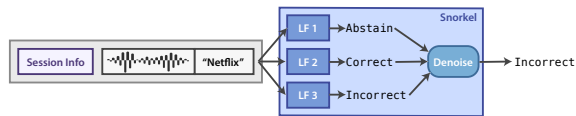


Figure 1: An example weak labeling. In this case, we would discard the incorrect transcript, “Netfix.”

Concretely, we feed an audio amplitude sequence $(x_t)_{t=1}^{\ell} \in [-1, 1]$ into a pretrained model consisting of one-dimensional convolutional feature extractors and transformer layers, getting frame-level context vectors $(\mathbf{h}_t)_{t=1}^N \in \mathbb{R}^k$. On each of these vectors, we perform a softmax transformation across the vocabulary V , for a final probability distribution sequence of $(\mathbf{y}_t)_{t=1}^N \in \mathbb{R}^{|V|}$. For fine-tuning, we use a training set composed of audio–transcript pairs and optimize with the standard connectionist temporal classification objective (CTC; Graves, 2012) for speech recognition. We uncase the transcripts and encode them with a character-based tokenizer, as is standard. At inference time, we decode the CTC outputs with beam search and a four-gram language model.

2.2 Data Curation

To build a weakly labeled dataset, we turn to Snorkel (Ratner et al., 2017), a popular data programming framework for aggregating and denoising weak labelers. In Snorkel, domain experts first create handwritten weak labelers, which the authors call labeling functions (LFs). Each of these LFs takes as input an unlabeled example, as well as any auxiliary data, and either outputs a label or abstains. Next, Snorkel applies these LFs to each example in a dataset, producing a matrix of noisy labels. It learns from this noisy observation matrix a generative model with the true labels as latent variables, which it supplies to downstream tasks.

Our task is to remove incorrect transcripts from a weakly constructed dataset. Our LF inputs are audio clips and transcripts, along with session data, and our outputs are one of correct, incorrect, or abstain. After Snorkel denoises the LF outputs and labels each dataset example, we discard abstained or incorrect ones, as visualized in Figure 1. We derive and use the three following novel LFs:

Session position. We group queries in the same session if each occurs within 60 seconds of at least one other and is issued by the same user. Previously, we found a negative correlation between the intrasession position of a query and the word-error

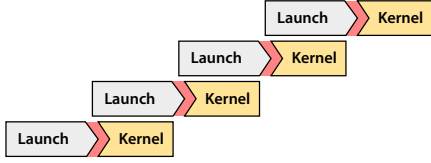


Figure 2: Typical way for the CPU to launch a sequence of small GPU kernels, with time flowing from left to right. Red area denotes launch latency.

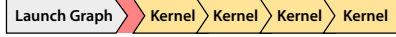


Figure 3: Launching a CUDA graph. Difference in right margin relative to Figure 2 portrays time savings.

rate (Tang et al., 2019b), where the last query consistently has a low word-error rate (WER), and long sessions have high intermediate query WERs. With this finding, we write the session position LF, given query q , as

$$\text{LF}_{\text{SP}}(q) := \begin{cases} \text{CORRECT} & \text{if } q \text{ is last in its session} \\ \text{INCORRECT} & \text{if sess. length} \geq 3, q \text{ not last} \\ \text{ABSTAIN} & \text{otherwise.} \end{cases}$$

ASR confidence. For each transcribed utterance, ASR systems output a confidence score, which correlates with the WER. In most systems, this score results from an addition between the acoustic model score and the language model score. The first is a function of speech, while the second of text. Since our third-party ASR service is opaque, we have access only to the final score. This complicates its direct use because thresholding it would skew the balance toward frequent words, as influenced by the language model.

To bypass this issue, we collect sample statistics of the final score *grouped by transcript text*, then design an LF with transcript-specific thresholds. This way, we remove the language model score as a confounder. Define

$$\text{LF}_{\text{AC}}(q) := \begin{cases} \text{CORRECT} & \text{if } s(q) \geq \text{p80}(q) \\ \text{INCORRECT} & \text{if } s(q) \leq \text{p20}(q) \\ \text{ABSTAIN} & \text{if } \text{p20}(q) \text{ or } \text{p80}(q) \text{ undefined} \\ & \text{or otherwise,} \end{cases}$$

where $s(q)$ is the confidence score for query q from the third-party ASR, and $\text{p20}(q)$ and $\text{p80}(q)$ return the 20th and 80th percentile ASR score for the transcript of q , respectively.

Rapid repetition. Users often rapidly repeat their voice queries upon ASR mistranscription (Li and Ture, 2020). Given this, we can discard queries that closely precede others from the same user:

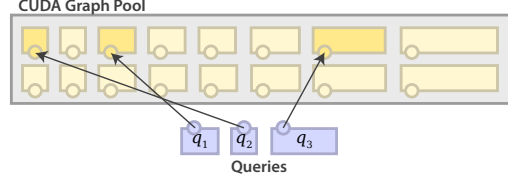


Figure 4: Three queries routed across a graph pool.

$$\text{LF}_{\text{RR}}(q) := \begin{cases} \text{INCORRECT} & \text{if the user's next query} \\ & \text{occurs } \leq 13 \text{ seconds of } q \\ \text{ABSTAIN} & \text{otherwise.} \end{cases}$$

On our platform, we’ve determined 13 seconds to be the optimal duration in terms of specificity and sensitivity (Li and Ture, 2020).

2.3 Model Inference Acceleration

In production, we use a batch size of one for inference. This largely decreases efficiency because GPU kernel launches now dominate the processing time, as portrayed in Figure 2. In our case, we can’t just pad to a large fixed size, since computation increases quadratically with length for transformers. It’s also infeasible to use batching (e.g., batch together sequential queries) because only 4–6 queries arrive in a 50-millisecond window per server, and we can’t afford to sacrifice that much speed.

To improve inference efficiency, CUDA graphs allow a sequence of GPU kernels to be captured and run as a single computation graph, thus incurring one CPU launch operation instead of many—see Figure 3. However, these graphs are input shape and control flow static, so they must be pre-constructed. This clearly poses a barrier to using variable-length audio as input.

To address this issue, we propose to allocate a pool of differently sized CUDA graphs, then route each query to the nearest upper-bound graph. For higher efficiency, we match the length distribution of the pool with that of the computation time on production traffic. Formally, let X be the random variable (r.v.) denoting the arrival distribution of the lengths of production-time queries. Let $Z := f(X)$ be the time it takes for a CUDA graph to perform inference for length X . Then, our CUDA graph pool comprises $\mathcal{G} := (g_{z_1}, \dots, g_{z_n})$, where g_{z_i} denotes a CUDA graph of length z_i and z_1, \dots, z_n are realizations of Z . To serve a query of length l , we pick the graph g_{z^*} , where

$$z^* := \min\{z_i \mid g_{z_i} \in \mathcal{G}, z_i \geq l\}. \quad (1)$$

Dataset	Train/Dev/Test Hrs.	# Speakers	# Unique
CC-20	22/2.2/2.2	40K/4K/4K	20
CC-LG	1400/1.0/2.5	325K/2K/4K	88K

Table 1: Dataset statistics. Further query distribution details are in the appendix.

Our upstream system sends no more than ten seconds of audio by design, bounding this set. We illustrate this process in Figure 4.

3 Experimental Setup

Our key experiments are to validate the model effectiveness of our labeling functions (Section 2.2) and the computational savings of our CUDA graph pool (Section 2.3). We trained every run on one p3.2xlarge Amazon Web Services (AWS) instance, which has an Nvidia V100 GPU and eight virtual CPU cores. We implemented our models in PyTorch using the HuggingFace Transformers library (Wolf et al., 2019) and Nvidia’s NeMo (Kuchaiev et al., 2019); see the appendix for more details.

3.1 Dataset Curation

We curated two datasets: one critical dataset, called CC-20, comprising the twenty most frequent commands, and another large-scale dataset, named CC-LG, consisting of audio examples sampled uniformly at random from user traffic. We split our datasets into one or more training sets, a development (dev) set, and a test set, all drawn from separate days and speakers—see Table 1 for statistics. On CC-20, native English speakers annotated the training set to establish an “upper bound” in quality, relative to using the weakly labeled datasets. On CC-LG, the 1400-hour set was too large to annotate, so we skipped that. On both datasets, we manually annotated the dev and test sets to serve as gold evaluation sets.

For the weakly labeled training sets, we constructed one set with raw transcripts from the third-party ASR system and another set with transcripts from Snorkel, filtered using the labeling functions in Section 2.2. We name the former set “raw” and the latter “weak.” To remove dataset size as a confounder, we use the same size for all training sets.

3.2 Baselines and Models

For our first baseline, we picked Google Cloud’s public ASR offering (Beaufays, 2022), primarily

Model	Training	CC-20	CC-LG
		Dev/Test	Dev/Test
Google Cloud	–	24.7/24.7	26.5/25.5
Our Third Party	–	7.56/7.60	10.8/9.66
Our Trained Models			
SEW _{tiny} 41M parameters	Raw	6.72/6.82	17.4/16.3
	Weak	5.17/4.80	15.9/14.5
	Human	4.79/4.66	–
Wav2vec2.0 _{base} 94M parameters	Raw	2.81/3.17	10.2/9.11
	Weak	1.62/1.77	9.14/8.82
	Human	1.54/1.75	–
Conformer _{large} 120M parameters	Raw	3.52/3.68	12.6/10.6
	Weak	3.63/4.08	12.0/9.78
	Human	2.60/2.72	–

Table 2: Dev and test WERs of models trained on sets without LFs (raw), with LFs (weak), and with human annotations (human). Best results bolded.

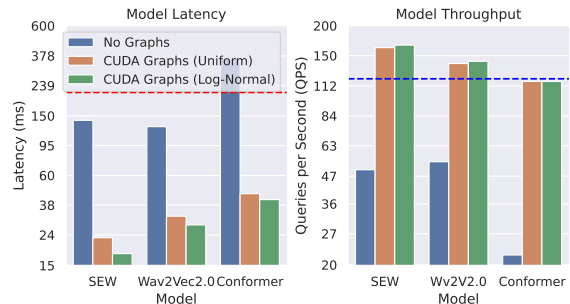


Figure 5: Throughput in queries per second and latency in milliseconds of all three models, under different CUDA graph pool settings. The red line on the left is our third-party ASR model latency and the blue line on the right our required throughput in production.

to sanity check our third-party ASR service. We used their standard model offering, touted as state of the art, costing us \$0.006 USD per 15 seconds of speech. For our second baseline, we selected our third-party ASR service that we licensed from a major American technology company.

Models. We chose three different state-of-the-art, pretrained transformer models from the literature, each representing a separate computational operating point: the Squeezed and Efficient Wav2vec model, tiny variant (SEW-tiny; Wu et al., 2022), at 41 million parameters; the standard Wav2vec 2.0 base model (Wav2vec 2.0-base; Baevski et al., 2020), at 94 million parameters; and the large Conformer model (Conformer-large; Gulati et al.,

Training Set	CC-20	CC-LG
	Dev/Test	Dev/Test
Raw (no LFs)	2.81/3.17	14.9/13.6
+ LF _{SP}	2.32/2.64	13.3/12.1
+ LF _{AC}	2.16/1.93	13.3/11.9
+ LF _{RR}	1.62/1.77	13.1/11.8
Human	1.52/1.75	–

Table 3: Quality of Wav2vec 2.0-base under differently constructed but equally sized training sets.

2020), at 120 million. We initialized them with LibriSpeech-fine-tuned weights and trained them using standard gradient-based optimization—we put details in the appendix.

4 Results and Discussion

We present our model quality results in Table 2. Unsurprisingly, Google Cloud does worse than our third-party service, which has been specifically tailored to our in-domain vocabulary. On average, sets curated with Snorkel (denoted as “weak”) improves the WER by 0.97 points (95% CI, 0.09 to 1.85) relative to those without (“raw”). Wav2vec 2.0-base, our best model, outperforms the third party by a relative 70% and 8% on CC-20 and CC-LG, respectively. Except for Conformer-large, all models trained on Snorkel-labeled sets achieve near parity with those on human-annotated training sets, with Wav2vec 2.0-base in particular reaching a test WER on CC-20 worse by only 0.02 points (1.77 vs. 1.75). We speculate that conformers perform worse than Wav2vec 2.0-base does due to using log-Mel spectrograms instead of raw audio waveform: our voice queries greatly differ in loudness, resulting in exponential fluctuations after applying the log transform (as the input approaches 0).

We chart our model acceleration results in Figure 5. We gather these statistics from replaying production-time traffic as fast as possible to saturate the model. Overall, CUDA graph pools accelerate our models by 7–9× (left subfigure; compare blue and green bars) and increase throughput by 3–5× (right subplot). Initializing the graph lengths to be log-normal distributed ekes out a few percentage points (compare orange and green) in performance, since that better matches our production traffic. Most stark is the contrast between vanilla, graphless conformer throughput (22 QPS) and its accelerated counterpart (117 QPS), representing a

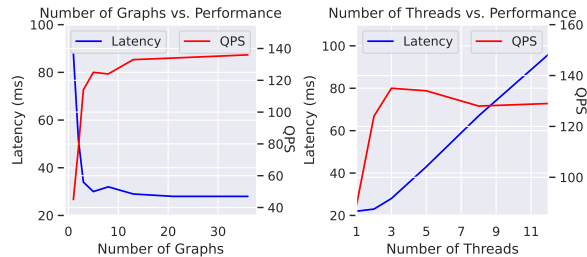


Figure 6: Twin plots of the system latency and throughput plotted against the number of CUDA graphs and inference threads, with the left y -axis tracking latency and the right axis throughput.

five-fold improvement. This likely arises from the vanilla conformer incurring much kernel launch overhead, on account of its more nested architecture, precisely which CUDA graphs address.

4.1 Ablation Studies

Data curation. We measure the quality contribution of each LF, as described in Section 2.2. We curate datasets using one additional LF at a time, starting with no LFs, then the session position LF, followed by the ASR confidence LF, and, finally, the rapid repetition LF. This process results in four datasets for the nested configurations. To remove transcript diversity and dataset size as confounders, we fix the number of training hours to 200 hours and match the transcript distributions. We target Wav2vec 2.0-base since it’s our deployment model.

We present the ablation results in Table 3. Each added LF improves the quality, with the first LF having the most impact (1.5 average points for the first vs. 0.1–0.7 for the rest), likely due to diminishing returns. We note that the ASR confidence score affects CC-20 more than it does CC-LG, possibly because of shorter sessions.

Model inference acceleration. We study how the number of CUDA graphs and inference threads (i.e., threads for launching graphs) affects the latency and the throughput, all else being equal. First, we sweep the number of CUDA graphs and hold the thread count at 3, the optimal value from our experiments. Next, we vary the thread count and fix the number of graphs at 36, also the best value. In both settings, we sample 10k queries uniformly at random from production and queue them up in our inference server, which comprises an Nvidia T4 GPU and an eight-core CPU.

We plot our results in Figure 6. For CUDA graphs, we observe rapidly diminishing returns

in both latency and throughput after 5–8 graphs, although they continue to improve until the final value of 36 graphs, the most we can fit in the GPU memory. For inference threads, we see initially rapid gains in throughput (though not latency) until 4 threads, whereupon throughput tapers slightly and latency grows linearly. We conjecture that this arises from GPU saturation causing thread contention; while we can certainly push more queries at a time (there being 36 graphs), the GPU can process only 138 queries worth per second. This results in a backlog of queries when we exceed 3–4 threads, causing linear growth in latency if throughput remains stable.

4.2 Industrial Considerations

We deploy SpeechNet as load-balanced Docker Swarm replicas, each exposing a WebSocket API for real-time transcription. We write the model server in Python and the inference decoder in C++; in particular, we free in the decoder Python’s global interpreter lock, a substantial bottleneck in our application. Our decoder runs faster than all tested open-source CTC decoders do, such as Parlance’s `ctcdecode`, `pyctcdecode`, and `Flashlight`. We execute all graphs in half-precision on separate CUDA streams, further increasing parallelism.

To monitor the reliability of our production system, we measure and expose four key service-level indicators (SLIs): query traffic, server errors, response latency, and system saturation. Taken together, these represent the so-called “Google Golden signals,” a battery of metrics espoused by its namesake. As is standard in industry, we export real-time metrics to Prometheus, a monitoring system for time series, and then aggregate them in Grafana, a full-stack visualizer.

During the initial release of SpeechNet, these metrics enabled us to detect and mitigate critical imperfections. In one such case, we observed a large spike in traffic preceding increases in time-out errors and latency. The spike occurred at the top of the hour, when, due to the nature of television programming, many users issue queries to change shows. From this evidence, we traced the culprit to our suboptimal decoder implementation, which we promptly fixed.

5 Related Work

Pretrained ASR models. Much like natural language processing, the dominant paradigm in the

end-to-end speech recognition literature is to pre-train transformers on vast quantities of unlabeled speech and then fine-tune on the labeled datasets. In their seminal work, [Schneider et al. \(2019\)](#) pioneer this approach with a contrastive learning objective, calling it Wav2vec. They further refine it in [Baevski et al. \(2020\)](#) by introducing discretized representations, naming their model the present Wav2vec 2.0. Other variants of this model include the Squeezed and Efficient Wav2vec model ([Wu et al., 2022](#)), which introduces architectural modifications for computational efficiency, and the conformer ([Gulati et al., 2020](#)), which adds convolutions in the transformer blocks for better local context modeling.

Weakly supervised ASR. Several papers explore constructing a weakly labeled dataset and training an ASR system with little to no human annotation. VideoASR ([Cheng et al., 2021](#)) and GigaSpeech ([Chen et al., 2021](#)) construct speech datasets from videos and subtitles, but this fails in our domain since our users’ voice queries differ greatly from those of public sources in both acoustics and text. For example, our queries contain rare entities (e.g., “Xfinity Home”), rarely last more than 4–5 seconds, and come from a low-fidelity microphone in frequently noisy households. Along a separate line, [Dufraux et al. \(2019\)](#) proposes a label noise-aware objective for ASR; however, this method increases training time by 15–30×, which is too burdensome for us.

Model acceleration. A plethora of model acceleration methods exist for transformers. In structured pruning, entire blocks of weights are removed, like attention heads ([Michel et al., 2019](#)) and weight submatrices ([Li et al., 2020](#)), resulting in a more lightweight model. This comes at the cost of quality, which we can’t sacrifice given our thin margin over our third party. [Hinton et al. \(2015\)](#) proposes knowledge distillation, where the outputs of a small model are fine-tuned against those of a large model, but we wish to use the original, pretrained model architecture at runtime for robustness. Still others propose low bit-width (2–8 bit) quantization ([Shen et al., 2020](#)), which, while quality preserving, has poor conventional GPU software support. Note that, in this paper, we restricted our experiments to CUDA graph pools because their application does not exclude others. In fact, when multiple acceleration methods can be applied, [Xin et al. \(2022\)](#) find that the savings are largely cumulative.

6 Conclusions and Future Work

In this paper, we explore commercializing a transformer-based, end-to-end speech recognition system without human annotation and with less computational power. We design three novel labeling functions, derived from implicit user feedback, for Snorkel to construct weakly labeled, in-domain speech datasets from production traffic. We also propose CUDA graph pools, a novel model acceleration method especially suited for single-example inference, as frequently encountered in production. Our system, SpeechNet, improves the word-error rate by a relative 8% and the inference speed by 600%, compared to our third-party ASR service. One promising research direction is to extend SpeechNet to the recently released OpenAI Whisper (Radford et al., 2022), an ultra large-scale ASR model trained on 680,000 hours of speech, representing the longest corpus to date.

Limitations

Our methods primarily apply to companies seeking to build out in-house ASR systems given at least a few thousand customers. We target business-to-consumer products, not business to business, where clients have wildly different needs without any guarantee on the userbase size (or even existence). Due to the setting of our work at a for-profit organization, we're also barred from releasing user data and source code out of concerns for privacy and intellectual property.

References

- Sherif Abdou and Michael S. Scordilis. 2004. Beam search pruning in speech recognition using a posterior probability-based confidence measure. *Speech Communication*.
- Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*.
- Françoise Beaufays. 2022. Google Cloud launches new models for more accurate speech AI. <https://cloud.google.com/blog/products/ai-machine-learning/google-cloud-updates-speech-api-models-for-improved-accuracy>.
- Guoguo Chen, Shuzhou Chai, Guanbo Wang, Jiayu Du, Wei-Qiang Zhang, Chao Weng, Dan Su, Daniel Povey, Jan Trmal, Junbo Zhang, et al. 2021. GigaSpeech: An evolving, multi-domain ASR corpus with 10,000 hours of transcribed audio. *arXiv:2106.06909*.
- Mengli Cheng, Chengyu Wang, Jun Huang, and Xiaobo Wang. 2021. Weakly supervised construction of ASR systems from massive video data. In *Proc. Interspeech 2021*.
- Adrien Dufraux, Emmanuel Vincent, Awni Hannun, Armelle Brun, and Matthijs Douze. 2019. Lead2Gold: Towards exploiting the full potential of noisy transcriptions for speech recognition. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.
- Alex Graves. 2012. Connectionist temporal classification. In *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer.
- Alan Gray. 2019. Getting started with CUDA graphs. <https://developer.nvidia.com/blog/cuda-graphs/>.
- Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech 2020*.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv:1503.02531*.
- Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al. 2019. NeMo: a toolkit for building ai applications using neural modules. *arXiv:1909.09577*.
- Bingbing Li, Zhenglun Kong, Tianyun Zhang, Ji Li, Zhengang Li, Hang Liu, and Caiwen Ding. 2020. Efficient transformer-based large scale language representations using hardware-friendly block structured pruning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.
- Wenyan Li and Ferhan Ture. 2020. Auto-annotation for voice-enabled entertainment systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in Neural Information Processing Systems*.
- Alec Radford, Jong W. Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust speech recognition via large-scale weak supervision. *OpenAI Blog*.
- Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the International Conference on Very Large Data Bases*.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. *Proc. Interspeech 2019*.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019a. Distilling task-specific knowledge from BERT into simple neural networks. *arXiv:1903.12136*.

Raphael Tang, Ferhan Ture, and Jimmy Lin. 2019b. Yelling at your TV: An analysis of speech recognition errors and subsequent user behavior on entertainment systems. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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*.

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:1910.03771*.

Felix Wu, Kwangyoun Kim, Jing Pan, Kyu J. Han, Kilian Q. Weinberger, and Yoav Artzi. 2022. Performance-efficiency trade-offs in unsupervised pre-training for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Ji Xin, Raphael Tang, Zhiying Jiang, Yaoliang Yu, and Jimmy Lin. 2022. Building an efficiency pipeline: Commutativity and cumulativity of efficiency operators for transformers. *arXiv:2208.00483*.

A Computational Environment

We train all models on Amazon p3.2xlarge instances running HuggingFace Transformers 4.15.0, from which we borrow the SEW and Wav2vec implementations; PyTorch 1.11.0 (CUDA 10.2), a popular deep learning framework; Nvidia’s NeMo library, which we depend on for the Conformer implementation; and SentencePiece 0.1.94, which we use for the character-based tokenizer. We implement our CTC decoder in C++14, interfacing with Python using pybind11 and the development libraries for SentencePiece and PyTorch (LibTorch). We serve users on geographically dispersed data

centers on the American east and west coasts, running Nginx-load-balanced boxes with Nvidia T4s.

B Dataset and Production Statistics

We curated CC-20 sampled across weeks of traffic, with training, dev, and test coming from separate speakers. We constructed CC-LG’s training set sampled from 2 days of traffic between July 3 and July 5, 2022 and the development/test sets from separate users sampled a day after the training set.

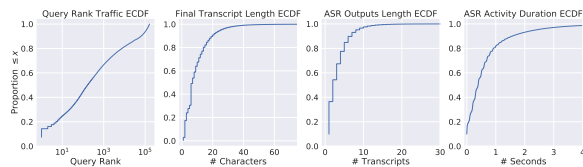


Figure 7: Distribution statistics of our in-production queries.

We present detailed production statistics of our queries in Figure 7. The query distributions have large right skew, with the top 1000 queries making up nearly 70% of the traffic, as the first subplot shows. Our queries are lexically simple, e.g., “Hulu,” “Free movies for me,” etc., as the second subgraph shows. The third and fourth subgraphs denote the activity of the ASR system—most queries are less than 1–2 seconds in speech (not necessarily total audio length).

C Training Details

For all models, on CC-LG, we first resize and re-initialize the final linear layer to match our vocabulary size, then fine-tune just the output linear layer (as recommended in the original Wav2vec paper) for 30k steps. Next, we ran 750k optimization steps on the “raw” training set. Then, we train for an additional 100k steps on the “weak” subset, if applicable. If it’s the raw training run, we still train for an additional 100k steps, but on the “raw” training set as usual. That is, all configurations on CC-20 use 850k optimization steps. On CC-20, we use 10k steps for the initial output layer fine-tuning and then ran 50k optimization steps for all models. We use the AdamW (Loshchilov and Hutter, 2018) optimizer with a batch size of 8 for all runs. We decode all model outputs using a beam size of 15 and a beam cutoff of 30. All model weights are initialized from the respective model cards on HuggingFace’s model zoo. We describe model-specific hyperparameters:

SEW. We optimize our models using a learning rate of 3×10^{-6} , determined from preliminary experiments across several choices spanning different orders of magnitude. SEW operates on the raw audio waveform.

Wav2vec 2.0. We use a learning rate of 2×10^{-6} , determined similarly. Wav2vec 2.0 operates on the raw audio waveform as well.

Conformer. As is standard, we transform all audio amplitudes to 80-dimensional Mel spectrograms before being input to the Conformer encoder. We pick a learning rate of 5×10^{-6} using the same procedure as the other models do.

Controlled Language Generation for Language Learning Items

Kevin Stowe*, Debanjan Ghosh*, Mengxuan Zhao

Educational Testing Service

{kstowe, dghosh}@ets.org, mzhao@etscanada.ca

Abstract

This work aims to employ natural language generation (NLG) to rapidly generate items for English language learning applications: this requires both language models capable of generating fluent, high-quality English, and to control the output of the generation to match the requirements of the relevant items. We experiment with deep pretrained models for this task, developing novel methods for controlling items for factors relevant in language learning: diverse sentences for different proficiency levels and argument structure to test grammar. Human evaluation demonstrates high grammatically scores for all models (3.4 and above out of 4), and higher length (24%) and complexity (9%) over the baseline for the advanced proficiency model. Our results show that we can achieve strong performance while adding additional control to ensure diverse, tailored content for individual users.¹

1 Introduction

Recent advancement of transformer (Vaswani et al., 2017) based pre-trained language models (LM) (Lewis et al., 2020; Brown et al., 2020; Raffel et al., 2020) have resulted in unprecedented success in generating large amounts of fluent English text. One possible area where text generation can be applied is item generation for English language learning applications (LLAs). LLAs are popular apps used by millions of people all over the world.² These apps often include multiple choice items for vocabulary tests, flashcards, grammar lessons, and more. Typically, such items are created manually (Service, 2010) or curated from crowd-sourced sentence database, e.g., Tatoeba (Settles et al., 2020).³

*Equal Contribution.

¹Code and datasets made available at <https://github.com/EducationalTestingService/concept-control-gen>

²<https://www.businessofapps.com/data/language-learning-app-market/>

³<https://tatoeba.org/en/>

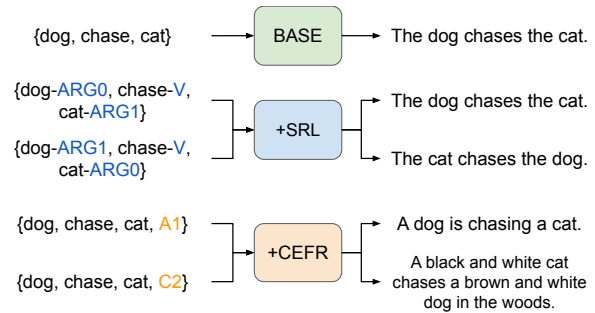


Figure 1: Controlling the concept2seq generation process, using semantic role labels and CEFR levels.

On the contrary, our goal is to make this process scalable by employing LMs, enabling developers of LLAs to be able to implement a much broader array of learning items quickly and efficiently.

This is accomplished by using a *concept2seq* framework: a sequence-to-sequence architecture in which, given a set of relevant *concepts*, we aim to generate sentences that minimally contain those concepts.⁴ There is a wide body of work relating to this framework (Lin et al., 2020; Carlsson et al., 2022) to generate sentences: we experiment with a number of adaptations that are applicable to LLAs. First, the importance of providing diverse content based on student/user needs and different skill sets is well established in learning sciences (Morgan, 2014; Clarke and Miles, 2003). We generate sentences according to different skill sets by conditioning on the Common European Framework of Reference for Languages (CEFR) levels in English.⁵ We use a document level CEFR predictor (Montgomerie, 2022) to predict the CEFR levels in the training data and in turn use the level in a controlled generation framework based on BART (Lewis et al., 2020) to generate different CEFR level-specific sentence (Section 3.1). Second, to

⁴Concepts are lemmatized tokens in text extracted by using the ConceptNet knowledge base (Speer et al., 2017).

⁵CEFR is an international standard for measuring user's ability within a language.

test the grammatical proficiency of the users we attach syntactic and semantic information in the form of semantic role labels (SRL) to the concept inputs for controlled generation where we can specify the semantic roles (e.g., ARG0) in the generation (Section 3.2). Such generations can be used in LLAs to ask grammar questions and further expand the diversity of items.

Consider the examples in Figure 1. The BASE model generates a sentence using the concepts “dog”, “chase”, and “cat”. The CEFR model demonstrates two different generations for two skill levels: a simple short sentence (e.g. “a cat is chasing a dog”) for the A1 beginner level and a long complex sentence (e.g., “a black and white . . . in the woods) for the C2 proficiency level (Section 3.1). Likewise, the SRL model presents examples where we conditioned the generations on specific semantic roles (Section 3.2).

We evaluate our models using automatic metrics relevant to our goals (perplexity, concept coverage in the generated sentences, length, and lexical diversity), as well as utilizing Amazon Mechanical Turk (MTurk) to get human judgments of important factors: grammaticality, complexity, and semantic plausibility. The CEFR model generates less complex, shorter sentences than the baseline when targeting the A1 level (complexity score of 2.45, average length of 11.6) compared to the C2 level (complexity score of 2.73, average length of 15.3 words). The SRL model generates the targeted words in the correct argument slot significantly more than the baseline (improving from 6% to 32% based on the targeted role). All models are within 3% of the baseline in terms of grammaticality and semantic plausibility, indicating that we can effectively generate sentences from concepts while adding additional control.

2 Data

We employ two datasets for concept2seq generation. Each instance in the datasets is a set of concepts paired with a sentence that contains those concepts. First, we use the COMMONGEN data which is based on existing caption corpora (Lin et al., 2020). From this dataset we use 71,408 concept/sentence pairs. Although COMMONGEN is used in related concept2seq generation (Lin et al., 2020), since it is based on image captions, many samples are phrases (not sentences) and less diverse. Thus, we also collect another dataset based on fourteen rel-

evant vocabulary items for language learning that belong to different CEFR levels.⁶ Sentences are collected from diverse sources such as the ROCStories (Mostafazadeh et al., 2016), Tatoeba sentence database, and the Google book corpus.⁷ We first retrieved sentences containing the vocabulary items from these different sources and then extracted the concepts using the ConceptNet knowledge-base by employing Becker et al. (2021). We extract noun, verbs, and adjective tokens as concepts and keep only those sentences containing 2-5 concepts to keep consistency with COMMONGEN. This dataset is denoted as the VOCABULARY dataset and consists of an additional 218,997 concept/sentence pairs. In total, the COMMONGEN and VOCABULARY gives us a dataset of 290,399 pairs of concepts and sentences.

To evaluate our generation models, we create two test sets to evaluate two different scenarios. Our goal is to evaluate concept sets that occur frequently in the training data, as well those that occur rarely. In order to build these two types of test sets, we first generate the frequency counts of each concept over the entire dataset. We then calculate the frequency of a given concept set as the sum of the frequency counts of each concept within that set. We then sample 500 instances from both the COMMONGEN and VOCABULARY datasets from the top 10% highest frequency concept sets and 500 each from the bottom 10% frequency sets. We split the test data this way to evaluate two likely use cases. The lowest 10% aligns with the case where we have unseen concepts and want to generate something novel; the most frequent 10% matches the everyday use case where we generate from things we’ve seen before. This gives us a test set of 2000 total sample, half from COMMONGEN and half from VOCABULARY, additionally split into high frequency and low frequency concepts. From the remaining dataset we randomly use 90% as training and 10% as validation.

3 Methods

Here, we present our computational approaches for sentence generation using the concept2seq framework. At its base form, our task is to generate a sentence s that consists of sequence of tokens, $s = \{s_1, \dots, s_m\}$ using a list of concepts

⁶This vocabulary list and their word derivatives was recommended by the learning scientists of the LLA.

⁷<https://www.english-corpora.org/googlebooks/>

$c = \{c_1, \dots, c_n\}$. Using the standard autoregressive sequence-to-sequence architecture (Sutskever et al., 2014) we model $P_\theta(s | c)$ as follows:

$$P_\theta(s | c) = \prod_i P_\theta(s_i | s_1, \dots, s_{i-1}, c) \quad (1)$$

Note, the resulting sentence s should contain relevant vocabulary from the concept set c , but is otherwise unconstrained. We use a pretrained BART model (Lewis et al., 2020) composed of a bidirectional encoder and an autoregressive decoder. In our simplest setup (called BASE), the input for BART is a set of concepts c and the output text s is a sentence that contains those concepts. For the BASE model, we fine-tune the `bart-base` model on our training data described above, and then generate sentences based on the test concepts. Note, this is equivalent to the BART based experiment reported in Lin et al. (2020).⁸

3.1 CEFR-controlled generation

Research in language learning has shown that students’ retention of words and texts increases when they encounter increasing diversity in content (Adelman et al., 2006; Frances et al., 2020). Since language learning is affected by many variables (Oxford and Nyikos, 1989), we focus on a specific variable - the CEFR levels - which are an international standard that measures text complexity and has strong correlations with learners’ skill sets and language learning ability (Papageorgiou et al., 2015). To this end, we generate sentences guided towards different skill sets of users by conditioning on the CEFR levels. These labels are, in increasing order of proficiency: A1, A2, B1, B2, C1, and C2, where A1 denotes a beginner level and C2 denotes high proficiency. Our goal is to be able to start with a list of concepts and generate a sentence at the appropriate proficiency level. We first use a document level CEFR predictor (Montgomerie, 2022) to predict the CEFR levels for each sentence in the training data. This tagger, which functions by combining lexical, syntactic, and other attested proficiency features, provides a tag from A1 to C2 for each sentence in the training dataset. In turn, we use this predicted CEFR level as control codes to guide sentence generation.

⁸Although Lin et al. (2020) also reported experimental results using other transformer-based LMs such as GPT-2 (Radford et al., 2019) and T5 (Raffel et al., 2020), we notice BART performs better on several metrics, so we continue to use BART.

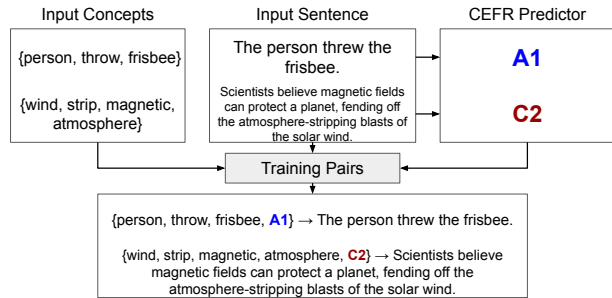


Figure 2: Method for applying CEFR labels to input concepts using the CEFR scorer.

Controlled generation models (Kikuchi et al., 2016; Hu et al., 2017; Fidler and Goldberg, 2017; Tsai et al., 2021) condition on a control code f in addition to the input c to model the distribution of $P_\theta(s | c, f)$. Similar to Eq. (1), we can write,

$$P_\theta(s | c, f) = \prod_i P_\theta(s_i | s_1, \dots, s_{i-1}, c, f) \quad (2)$$

Text generation conditioned on such control codes, such as sentiment control of movie reviews, style for chatbots, diverse story continuations, question generation etc., have been used effectively in recent research (Tu et al., 2019; Krause et al., 2021; Roller et al., 2021; Gao et al., 2022). We use the same idea for sentence generation by conditioning on the CEFR levels. Figure 2 shows the overview of the process.

3.2 Argument Structure-controlled generation

As the second task, we use the argument structure of the generated sentences as the control code. We determine for any given concept in the input what semantic role that concept should play in the output text. This gives two key advantages: we can ensure the semantic viability of generations in which the concepts make more sense in particular roles, and we can extend the variety of sentences generated by varying the semantic roles of the arguments. Consider the following generation:

{dog, chase, cat} → (a) the dog chased the cat
(b) the cat chased the dog

As the concepts are unordered, the model can generate both sentences where (a) the dog is chasing the cat and where (b) the cat is chasing the dog. Stereotypically, we would expect (a), but (b) is a viable reading. By enforcing the semantic roles of the concepts, either with the dog or the cat as the agent, our aim is to be able to more concretely

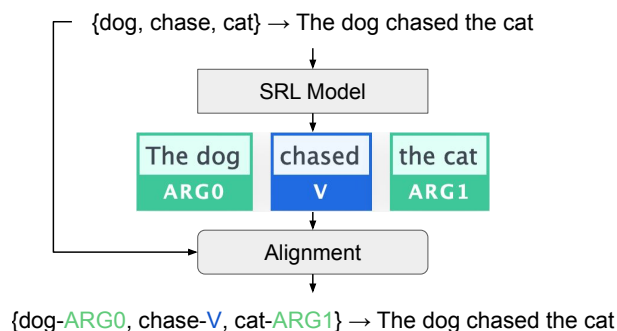


Figure 3: Method for applying SRL parse to the original concepts during training.

choose which output we’d like to see. This in turn can be utilized to check grammar skills of the users by follow-up questions in an LLA (e.g., which is the agent in the sentence?).

To identify the semantic roles, we tag the training dataset with an automatic semantic role labeling system (Stanovsky et al., 2018).⁹ For each verb in the input, the system tags each word in the sentence with the argument it takes in that verb’s scope. In order to convert these tags to control codes, we first extract each word in the sentence that matches a lemmatized version of the one of the input concepts. For each of these words, we identify all of the possible roles it can play in a sentence (note that words can take multiple roles, when they are arguments of separate verbs). We then iterate through these options, aligning the possible semantic role labels that word can take to the concepts in the input. This yields a new batch of concepts labeled with semantic role information that serve as the inputs for the given sentence. An overview of this process is shown in Figure 3.

Inference For CEFR-controlled generation, we have trained a single model on the full scope of CEFR tagged data: in order to generate sentences at a particular level, we need to provide that level to the model at inference time. For this, we experiment with the simplest level (A1) and the most advanced level (C2). We add these labels to the concept inputs to generate sentences that should match those levels: these setups are dubbed CEFR-A1 and CEFR-C2. provided different CEFR levels to generate different sentences. For argument structure-controlled generation, we run our SRL tagger on the test data and then apply these to the input sources to generate a SRL-controlled output, as was done in training (Figure 3). We generate

⁹<https://github.com/allenai/allennlp>.

using top-K sampling ($k = 50$) with a maximum length of 64 and a length penalty of 1.0.

3.3 Additional pretraining

The above BART model is originally trained with text as both the input and output. Our task is somewhat different, as the input consists of concepts. While these concepts are superficially a set of keywords, this still differs from what the original BART encoders have seen during training. In order to encourage the model to better handle this concept2seq data formulation, we leverage the power of additional pretraining, which has been shown to further improve model performance on new tasks (Gururangan et al., 2020).

We perform additional pretraining using wikipedia. Starting with a dump of wikipedia data,¹⁰ we first extract sentences which are then run through our concept extraction pipeline (Section 2). We filter down to 10M random concept-sentence pairs with 2-5 concepts/sentence. These 10M pairs are then used to continue training on top of the pretrained BASE model (the WIKI model).¹¹

4 Results

Evaluating natural language generation tasks can be difficult, and some automatic metrics can be problematic (Reiter, 2018). To overcome these difficulties, we use metrics specifically tailored to our task, as well as performing manual evaluation to get a concrete understanding of model performance.

4.1 Automatic Evaluation

Standard metrics, e.g., BLEU (Papineni et al., 2002) or ROUGE-L (Lin, 2004) that are often used to evaluate NLG outputs require true reference sentences for evaluation purpose. These methods are insufficient for our approach – our goal here is to generate sentences containing particular concepts conditioned on specific controls (e.g. CEFR) – and the resulting outputs do not need to match any particular gold standard. For that reason, we employ the following reference-free metrics for evaluation.

First, **perplexity** under a language model can indicate the fluency of the text. We report average perplexity per word using the GPT-2-base LM (Radford et al., 2019) in the generated sentences. **Coverage** indicates whether the generated

¹⁰<https://dumps.wikimedia.org/enwiki/latest/>

¹¹Full details of the training procedures are in Appendix A.3.

Model	Perplexity	Coverage (All)	Coverage (Any)	Length	Diversity
BASE	4.51	50.55	93.34	12.13	43.34
SRL	4.53	51.40	94.10	11.98	43.51
CEFR (A1)	4.52	49.16	92.70	11.58	43.95
(C2)	4.41	39.70	74.61	15.26	37.31
WIKI	4.58	51.51	94.10	12.28	42.88

Table 1: Automatic evaluation of generation models. Lower scores indicate better **Perplexity** and **Diversity**.

Model	V	ARG0	ARG1	ARGM
BASE	88.18	70.48	73.34	58.67
SRL	94.06	88.93	88.01	77.60

Table 2: Automatic evaluation of SRL coverage. Scores refer to percentage of times the label occurred with a given concept in the output over the input.

sentences contains the input concepts. We evaluate the percentage of generations that contain lemmas matching the input concepts in two ways: first, the percentage of outputs containing **any** lemmas matching the input as well as the percentage of those where **all** of the concepts are found in the output. We also measure average **length** of the generated sentences (in number of words). Finally, we measure lexical **diversity**: for this, we use the average tf-idf score of all non-stopwords in the sentence (learned from a recent Wikipedia dump): higher scores indicate more common words, while lower scores indicate more lexical diversity. Relevant results are shown in Table 1.

We note a number of observations from automatic metrics: perplexity remains relatively stable across models, indicating they all can produce fluent sentences. CEFR C2 has the lowest perplexity, indicating that BART can produce complex but still fluent sentences. Coverage, length, and diversity remain relatively stable across models as well. One exception is the CEFR C2 model, which has lower coverage (39.70) and higher length (15.26 words per sequence). Since C2 sentences in the training dataset are longer (averaging 25.1 words per sentence, compared to the overall average of 17.0), it is expected that CEFR C2 model produce longer sentences when higher proficiency is required. Likewise, the CEFR A1 model tends towards shorter sentences (11.58 words per sequence). Finally, the low diversity score of the CEFR C2 model indicate the complex sentences generated by the model have higher lexical diversity.

SRL Overlap Evaluation: Note that for the SRL model, we evaluate the test data with a single

set of SRL labels generated from the original SRL model. There are many ways to apply SRL labels to a given set of concepts, and we only evaluate against a single reference.

We use an automatic parser to capture whether the SRL-based inputs are accurately represented in the outputs. For the four most frequent argument types (ARG0, ARG1, ARGM, and V), we evaluate accuracy by comparing its presence in the generated output to its presence in the control codes. We measure the percentage of times the argument type is correctly represented by a concept in the generated sentence over the number of times the instructions indicate it should be. We compare the SRL to the BASE model in Table 2.

We might expect a large, pretrained model like BART to automatically generate the concepts into their expected roles, but we can see that the BASE model actually fluctuates greatly: for non-verbal arguments, it generates them in the semantic role of the reference sentence less than 75% of the time. This isn’t necessarily a problem, as the system still is generating sentences with the appropriate concepts, but it highlights the usefulness of argument control: using the SRL model, we can generate concepts into specific semantic roles much more consistently, with scores ranging from 77% to 94%, thus, improving by a large margin over the BASE model.

4.2 Human Evaluation

For human evaluation, we aim to capture three essential criteria that are important for test item generation. These are:

1. **Grammaticality:** The generated sentences/phrases should be grammatical, and should follow normal English syntax.
2. **Complexity:** The complexity of a given sentence as it relates to end users.
3. **Plausibility:** The generated items should describe semantically plausible scenarios, or they

Model	Grammatical		Complex		Plausible	
	Common	Rare	Common	Rare	Common	Rare
BASE	3.50	3.40	2.41	2.58	3.60	3.61
SRL	3.48	3.46	2.25	2.45	3.58	3.61
CEFR (A1)	3.50	3.44	2.40	2.50	3.63	3.61
(C2)	3.31	3.41	2.67	2.78	3.57	3.61
WIKI	3.44	3.42	2.46	2.60	3.56	3.59

Table 3: MTurk evaluation for each generation model, evaluated for grammaticality, complexity, and plausibility. *Italic* scores are significantly below the baseline; **bold** are significantly above (two-tailed t-test, $p < .01$).

Input Concepts	Model	Sentence	Gram	Comp	Plaus
{competition, agree, strange, participate}	BASE	I agree with you that it’s strange to participate in a competition with your dog.	4	3	4
	SRL	They agreed to participate in the strange competition.	4	2	4
	CEFR (A1)	I agreed to participate in the strange competition.	4	2.67	3.67
	CEFR (C2)	In some cases, it might be safer to think of different words to use if you don’t want to participate in a strange competition.	4	3	3.67
	WIKI	In a strange twist of luck, i’d like to participate in a competition. I don’t agree with you.	2	3.33	2.33
{girl, clothes}	BASE	A girl in black clothes is standing on a ladder and drawing on a wall.	4	2.33	4
	SRL	A girl in black clothes is playing soccer.	4	2.33	4
	CEFR (A1)	A girl in black clothes is playing with a toy lawn mower.	4	1.67	4
	CEFR (C2)	Young people sit in plastic chairs arranged around a set of stairs in a covered concrete area, wearing swim clothes, resting, and waiting.	4	3	4
	WIKI	A girl in black clothes playing soccer.	3.33	2	4

Table 4: Examples of generated sentences via different models with their annotated scores.

risk confusing or even misinforming the user.

We evaluated the generated outputs across these three criteria using Amazon Mechanical Turk (MTurk). We evaluated 800 sentences generated from each model. Three crowd-annotators were employed for each task and were asked to evaluate each sentence on a four-point scale for each criteria. We included many examples in the instructions. Each Human Intelligence Task (HIT) contained ten sentences to judge and we paid \$2 per HIT. We obtain three scores for each of the above criteria for each generated sentence, and take their mean as the final score (Table 3).

We observe a number of key take-aways from the human evaluations. First, the rare concept sets are more likely to yield more complex generations, but otherwise they are fairly similar to the common sets: they exhibit similar grammaticality and plausibility scores. All models score strongly for grammaticality: the CEFR C2 model is lowest, as it is attempting to generate more complex sentences and likely to make more mistakes, but all models average about 3.4. Second, with regard to complexity, the CEFR A1 model scores lower than the BASE while the CEFR C2 model scores higher: this

is our expected result, as the lower A1 level instruction yields simpler sentences, while the higher level C2 yields more complex sentences. Third, all models perform similarly with regard to plausibility, with every model being within .04 of the baseline. Finally, we see that additional pretraining doesn’t improve performance significantly over the baseline: the BART-base model seems perfectly capable of adapting to concept2seq instructions without additional pretraining.

Table 4 presents two examples from our models along with average human ratings for all three aspects. In general, CEFR C2 has produced long sentences with high complexity for all examples. Likewise, grammaticality and plausibility scores are almost perfect except one example from WIKI.

In general, the methods we implemented to allow for additional control (CEFR and SRL) function as expected: we can manipulate the proficiency and argument structure of the generated sentences to a significant degree, allowing us to develop diverse content for users at different levels for LLAs.

4.3 Performance Time

The model was trained on a single nVidia K80 GPU for approximately 158 minutes, at approximately 81 training samples processed per second. We are then able to generate approximately 54 sentences per second at inference time. While this makes the system capable in some regards of generating live learning items, this is not desirable nor is it our use case. There are substantial risks involved in generating items live and presenting them to users, including possible grammatical and semantic disfluencies, unsuitable content, and biases inherent in generation from language models (Sheng et al., 2021). Rather, this system is designed to be run offline, generating a batch of possible learning items that can then be curated by experts.

5 Related Work

The concept2seq generation problem has been investigated in several recent studies. Lin et al. (2020) released the COMMONGEN dataset and generated sentences using various transformer models, (Carlsson et al., 2022) have proposed prompting for generation, and (Zhou et al., 2020) have conducted instruction tuning for generation using concepts. Our work is related to the above and our novelty is that we utilize this framework to generate LLA items. Although we did not experiment with the ordering of concepts similar to (Zhao et al., 2022), our SRL based generation in fact implicitly control the order of the concepts by offering specific grammar roles.

In prior work on controllable generation, embedding vectors of the control variables were fed into the model to control the output (Kikuchi et al., 2016; Fan et al., 2018), whereas our approach resembles recent efforts where the control variable is concatenated to the main input (Keskar et al., 2019) to control particular style, such as sentiment, style for chatbots, diverse story continuations and argument generation (Tu et al., 2019; Schiller et al., 2021; Krause et al., 2021; Roller et al., 2021).

6 Conclusion and Future Work

We proposed a type-controlled sentence generation framework for LLAs. We generate sentences (a) conditioned on the CEFR levels to provide content for users/students who belong to different skill sets (e.g., beginner or proficient in English), and (b) conditioned with specific argument structures for grammar. In automatic evaluation, the SRL model

shows better coverage of input concepts than BASE, whereas human evaluation demonstrates high grammatically scores (3.4 and above) for all the models as well as high complexity for the CEFR C2 model that was designed to generate complex sentences for proficient users. In future, we want to continue a couple of error analyses on the input as well as on the generated sentences. Having taken into account that input data is pre-processed in several ways (e.g., concept extraction (Becker et al., 2021) and analysis of semantic roles (Stanovsky et al., 2018)), we want to select a small subset of data to determine whether such extraction has any error. Likewise, we also want to employ expert content developers to analyze the results of the CEFR predictor. Finally, we plan to employ additional controls such as word senses to guide context specific generations.

Acknowledgments

Thanks to Casey Medlock Paul and Kristen Herrick for suggesting reference materials on learning science, as well as Swapna Somasundaran for helpful comments.

7 Ethical Considerations

We leverage the freely available COMMONGEN dataset for model training. Though we have not exhaustively checked the dataset, given COMMONGEN is based on a variety of caption datasets, we consider them relatively safe and do not find any objectionable content. Likewise, we create another dataset, VOCABULARY, which is based on standard narratives and sentence databases that are used in many recent work. Training is done using large pretrained models that have been shown to have bias. Although the generated content do not appear biased, they may hallucinate content, which is a common problem for neural generation models. In future work, we plan to analyze and identify hallucinations from the generations, and assess possible bias issues within these generations.

Finally, we obtained institutional review board permission to conduct MTurk based evaluations to collect judgments from crowd workers regarding the quality of the sentences.

References

James S Adelman, Gordon DA Brown, and José F Quesada. 2006. Contextual diversity, not word frequency,

- determines word-naming and lexical decision times. *Psychological science*, 17(9):814–823.
- Maria Becker, Katharina Korfhage, and Anette Frank. 2021. **COCO-EX: A tool for linking concepts from texts to ConceptNet**. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Fredrik Carlsson, Joey Öhman, Fangyu Liu, Severine Verlinden, Joakim Nivre, and Magnus Sahlgren. 2022. **Fine-grained controllable text generation using non-residual prompting**. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6837–6857, Dublin, Ireland. Association for Computational Linguistics.
- John Clarke and Sherri Miles. 2003. Changing systems to personalize learning: Introduction to the personalization workshops. Technical report, The Education Alliance at Brown University.
- Angela Fan, David Grangier, and Michael Auli. 2018. **Controllable abstractive summarization**. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia. Association for Computational Linguistics.
- Jessica Fidler and Yoav Goldberg. 2017. **Controlling linguistic style aspects in neural language generation**. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark. Association for Computational Linguistics.
- Candice Frances, Clara D Martin, and Jon Andoni Duñabeitia. 2020. The effects of contextual diversity on incidental vocabulary learning in the native and a foreign language. *Scientific reports*, 10(1):1–11.
- Lingyu Gao, Debanjan Ghosh, and Kevin Gimpel. 2022. “what makes a question inquisitive?” a study on type-controlled inquisitive question generation. In *Proceedings of the 11th Joint Conference on Lexical and Computational Semantics*, pages 240–257, 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 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.
- Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. **Toward controlled generation of text**. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR.
- Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. <https://arxiv.org/abs/1909.05858>. ArXiv:1909.05858.
- Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. **Controlling output length in neural encoder-decoders**. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas. Association for Computational Linguistics.
- Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. **GeDi: Generative discriminator guided sequence generation**. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- 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.
- Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. **CommonGen: A constrained text generation challenge for generative commonsense reasoning**. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online. Association for Computational Linguistics.
- 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.
- Adam Montgomerie. 2022. CEFR english level predictor. <https://github.com/AMontgomerie/CEFR-English-Level-Predictor>.
- Hani Morgan. 2014. Maximizing student success with differentiated learning. *The Clearing House: A Journal of Educational Strategies, Issues and Ideas*, 87(1):34–38.
- Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende,

- Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849.
- Rebecca Oxford and Martha Nyikos. 1989. Variables affecting choice of language learning strategies by university students. *The modern language journal*, 73(3):291–300.
- Spiros Papageorgiou, Richard J Tannenbaum, Brent Bridgeman, and Yeonsuk Cho. 2015. The association between toefl ibt® test scores and the common european framework of reference (cefr) levels. *Research Memorandum No. RM-15-06*. Princeton, NJ: Educational Testing Service.
- 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*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- 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*, 21:1–67.
- Ehud Reiter. 2018. **A structured review of the validity of bleu**. *Computational Linguistics*, 44(3):393–401.
- Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. **Recipes for building an open-domain chatbot**. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 300–325. Association for Computational Linguistics.
- Benjamin Schiller, Johannes Daxenberger, and Iryna Gurevych. 2021. **Aspect-controlled neural argument generation**. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–396, Online. Association for Computational Linguistics.
- Educational Testing Service. 2010. Toefl ibt® test framework and test development. *TOEFL iBT Research Insight*, 1.
- Burr Settles, Geoffrey T LaFlair, and Masato Hagiwara. 2020. Machine learning–driven language assessment. *Transactions of the Association for computational Linguistics*, 8:247–263.
- Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2021. **Societal biases in language generation: Progress and challenges**. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4275–4293, Online. Association for Computational Linguistics.
- Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*, pages 4444–4451.
- 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.
- Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. **Sequence to sequence learning with neural networks**. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.
- Alicia Y. Tsai, Shereen Oraby, Vittorio Perera, Jiun-Yu Kao, Yuheng Du, Anjali Narayan-Chen, Tagyoung Chung, and Dilek Hakkani-Tür. 2021. Style control for schema-guided natural language generation. <https://arxiv.org/abs/2109.12211>. ArXiv:2109.12211.
- Lifu Tu, Xiaohan Ding, Dong Yu, and Kevin Gimpel. 2019. **Generating diverse story continuations with controllable semantics**. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 44–58, Hong Kong. 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**. 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.
- 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.

Chao Zhao, Faeze Brahman, Tenghao Huang, and Snigdha Chaturvedi. 2022. [Revisiting generative commonsense reasoning: A pre-ordering approach](#). In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1709–1718, Seattle, United States. Association for Computational Linguistics.

Wangchunshu Zhou, Dong-Ho Lee, Ravi Kiran Selvam, Seyeon Lee, Bill Yuchen Lin, and Xiang Ren. 2020. Pre-training text-to-text transformers for concept-centric common sense. <https://arxiv.org/abs/2011.07956>. ArXiv:2011.07956.

A Appendix

A.1 Vocabulary items

The following fourteen words in Table 5 (see their associated CEFR levels) were suggested by the learning scientists/content developers of the LLA we are involved with.

Word	CEFR
Clothes	A1
Famous	A1
Electric	A2
Return	A2
Lose	B1
Delicious	B1
Entertainment	B1
Literature	B1
Atmosphere	B2
Participate	B2
Awkward	B2
Solar	B2
Devote	B2
Caution	C1

Table 5: Words that are used to generate the VOCABULARY dataset (with their CEFR levels).

A.2 Concept Extraction

We extracted concepts using the concept extractor tool CoCo-Ex (Becker et al., 2021). The tool first parse sentences using standard parsers and then match tokens as found in ConceptNet knowledge base. The resulted concepts are categorized to their parts-of-speech. For our work we use nouns, verbs, and adjective tokens.

A.3 Model Training

We train our models using the HuggingFace platform (Wolf et al., 2020). We use the `bart-base` model as the initial checkpoint from the HuggingFace repository (Wolf et al., 2020). Each model is trained for 5 epochs with a batch size of 32 and a learning rate is $5e-5$, as these parameters yielded the best performance on the validation set. For the CEFR generation, the label is added as an additional concept. For the SRL-based generation, the labels are concatenated to individual concepts. For the additional pretraining with the Wikipedia data, we ran pretraining for 3 epochs. All experiments were conducted using NVIDIA K-80 GPUs.

A.4 MTurk Experiments

In order to collect human evaluation for generated sentences, we deployed our data collection pipeline using AWS infrastructures. After reading and confirming the consent page, Turkers are directed to the survey interface where detailed instructions and survey questions are presented (shown in Figure 4). Turkers must complete all questions to be able to submit. We initially began evaluation using 5 samples per HIT, but extended this to 10 as the time necessary for annotators to complete a HIT was extremely short. A “survey code” is returned to the Turker as the submission is successful, and with the code the Turker can submit the HIT and to be qualified for payment. All evaluation results are saved into a DynamoDB table through API Gateway and Lambda.

To ensure the quality of evaluation, we recruited Turkers with more than 10,000 approved HITs and the approval rate or 97% and above. One task is required to be completed within 60 minutes, and most Turkers were able to submit within 10 to 15 minutes.

A.5 Human Evaluation

Table 6 presents more examples of MTurk annotations.

Sentence Evaluation

Submit Survey
Survey Code :

c. Does the sentence/phrase seem plausible?

0: not possible *E.g. The person threw the frisbee to the moon.*

1: possible but extremely unlikely *E.g. The dog threw the frisbee to the person.*

2: somewhat unlikely *E.g. The person threw the frisbee to the cat riding the dog.*

3: normal, commonsense *E.g. The person threw the frisbee to the dog.*

When you have finished responding to all the questions,

- Enter your worker ID and click **Submit Survey** to generate a survey code.
- Submit this survey code via Amazon Mechanical Turk to be eligible for payment.

Sentence 1: The marketing campaign urges residents to participate in a two-week social distancing effort to halt the spread of the coronavirus.

Does the sentence/phrase seem grammatically correct?

0 1 2 3

Does the sentence/phrase seem complex?

0 1 2 3

Does the sentence/phrase seem plausible?

0 1 2 3

Sentence 2: A man wearing nothing but a cloth around his waist is standing, facing a crowd with a bright red substance on his back.

Does the sentence/phrase seem grammatically correct?

0 1 2 3

Does the sentence/phrase seem complex?

0 1 2 3

Does the sentence/phrase seem plausible?

0 1 2 3

Figure 4: The survey interface where Turkers can read detailed instructions and complete the evaluation of ten sentences.

Input Concepts	Model	Sentence	Gram	Comp	Plaus
{pace, author, contemporary, literature}	BASE	I'm not interested in literature	4	1.33	4
	SRL	I'm not interested in literature.	4	1.33	4
	CEFR (A1)	I'm not very interested in literature.	4	1.67	4
	CEFR (C2)	Although many european researchers have studied ancient persian literature in the nineteenth century, the new world is not paying attention to our contemporary literature.	3	3.33	2.67
	WIKI	He is very interested in literature.	4	2.33	4
{competition, agree, strange, participate}	BASE	I agree with you that it's strange to participate in a competition with your dog.	4	3	4
	SRL	They agreed to participate in the strange competition.	4	2	4
	CEFR (A1)	I agreed to participate in the strange competition.	4	2.67	3.67
	CEFR (C2)	In some cases, it might be safer to think of different words to use if you don't want to participate in a strange competition.	4	3	3.67
	WIKI	In a strange twist of luck, i'd like to participate in a competition. I don't agree with you.	2	3.33	2.33
{girl, clothes}	BASE	A girl in black clothes is standing on a ladder and drawing on a wall.	4	2.33	4
	SRL	A girl in black clothes is playing soccer.	4	2.33	4
	CEFR (A1)	A girl in black clothes is playing with a toy lawn mower.	4	1.67	4
	CEFR (C2)	Young people sit in plastic chairs arranged around a set of stairs in a covered concrete area, wearing swim clothes, resting, and waiting.	4	3	4
	WIKI	A girl in black clothes playing soccer.	3.33	2	4
{intersection, people, clothing}	BASE	A group of people in dark clothing are crossing an intersection in a city.	4	2	4
	SRL	A group of people in dark clothing are corssing an intersection in a city.	4	2	4
	CEFR (A1)	A group of people in heavy clothing congregat near an intersection.	3.667	2	4
	CEFR (C2)	A motorcycle is standing near a decaying building wall, window, and close-line with several clothing items hanging.	3.667	3.333	3.667
	WIKI	A group of people in reflective clothing is crossing an intersection together.	3.667	3	3.667
{stand, rock, water, body}	BASE	The man stands on the rocks near the body of water.	3.333	2.667	4
	SRL	A man stands on a rock near a body of water.	4	2.333	4
	CEFR (A1)	A body of water standing on rocks.	3.333	2	2
	CEFR (C2)	Young man standing on rocks near body of water in the mountains.	3.333	2.667	4
	WIKI	A man stands on a rock near a body of water.	4	2.333	4
{sand, beach, group, sit, large, clothing}	BASE	A man in black clothing and a black hat is walking in front of a yellow wall.	4	2.667	4
	SRL	A man in black clothing walking down a sidewalk.	3	2	4
	CEFR (A1)	A man in very torn, dirty clothing is walking down a city street in front of a well dressed woman.	4	3.333	4
	CEFR (C2)	A young man in colorful clothing and a helmet is walking down the street in front of a well dressed woman and young girl.	4	2.667	4
	WIKI	A woman wearing heavy makeup and a pink jacket is walking past a clothing shop at night.	4	3	4
{kosovo, game, participate}	BASE	In kosovo, they participate in a game of hopscotch.	3	2.333	4
	SRL	Kosovo will participate in the olympic games.	3.667	1.667	3.667
	CEFR (A1)	In 2006, the olympic games were cancelled after kosovo decided to participate in the oclolympic game.	3.667	2.667	1
	CEFR (C2)	The olympic athlete participates in his first match as a professional athlete in the olympics.	3.333	2	4
	WIKI	I will participate in the kosovo olympic games.	3.667	1.667	3.667

Table 6: Examples of generated sentences via different models with their annotated scores.

Improving Text-to-SQL Semantic Parsing with Fine-grained Query Understanding

Jun Wang, Patrick Ng, Alexander Hanbo Li, Jiarong Jiang,
Zhiguo Wang, Ramesh Nallapati, Bing Xiang, Sudipta Sengupta

Amazon AWS AI Labs

{juwanga, patricng, hanboli, jiarongj, zhiguow, rnallapa, bxiang, sudipta}@amazon.com

Abstract

Most recent research on Text-to-SQL semantic parsing relies on either parser itself or simple heuristic based approach to understand natural language query (NLQ). When synthesizing a SQL query, there is no explicit semantic information of NLQ available to the parser which leads to undesirable generalization performance. In addition, without lexical-level fine-grained query understanding, linking between query and database can only rely on fuzzy string match which leads to suboptimal performance in real applications. In view of this, in this paper we present a general-purpose, modular neural semantic parsing framework that is based on token-level fine-grained query understanding. Our framework consists of three modules: named entity recognizer (NER), neural entity linker (NEL) and neural semantic parser (NSP). By jointly modeling query and database, NER model analyzes user intents and identifies entities in the query. NEL model links typed entities to schema and cell values in database. Parser model leverages available semantic information and linking results and synthesizes tree-structured SQL queries based on dynamically generated grammar. Experiments on SQUALL, a newly released semantic parsing dataset, show that we can achieve 56.8% execution accuracy on WikiTableQuestions (WTQ) test set, which outperforms the state-of-the-art model by 2.7%.

1 Introduction

As a natural language interface to database, Text-to-SQL semantic parsing has made great progress in recent years with availability of large amount of annotated data and advances of neural models (Guo et al., 2019; Ma et al., 2020; Rubin and Berant, 2020; Wang et al., 2020; Zeng et al., 2020). These models typically employ a standard encoder-decoder modeling paradigm where model first encodes query and schema, then autoregressively decodes an executable program which could be a se-

quence of logical form tokens for flat decoding (Shi et al., 2020) or an abstract syntax tree (AST) for structured decoding (Lin et al., 2019; Wang et al., 2020). Either way, columns and tables are copied from input schema and literal values are copied from input query using pointer network in output program (Shi et al., 2020; Wang et al., 2020).

Despite the success of these models, there are several issues that are left unaddressed for real applications. **First**, without fine-grained query understanding, autoregressive top-down decoding suffers from generalizing to unseen query patterns at inference time (Herzig and Berant, 2020; Oren et al., 2020; Scholak et al., 2021), a problem commonly known as *compositional generalization*. Specifically, model may fail to synthesize correct SQL for compound input queries like “*where is the cyclist who has the most points from*” given training queries such as “*where is the runner from*”, “*which cyclist gets the most medals*”. **Second**, in previous works literal values in output logical forms are either omitted (Wang et al., 2020) or directly copied from input utterances (Brunner and Stockinger, 2020; Shi et al., 2020). The former will generate non-executable queries. The latter is problematic because mentions in query are often different from their canonical forms in database. For instance, assuming a query like “*how many points does LBJ get in last game*”, directly copying word “*LBJ*” into SQL query won’t match the name “*LeBron James*” in the database. **Third**, as is shown in Guo et al. (2019); Lin et al. (2019), structured decoding is effective in semantic parsing tasks as it’s more likely to generate coherent, executable SQL queries. In structured decoding, a sequence of production rules is generated from context free grammars along with schema and literal values. However, among existing solutions, some parsers require manually designed grammars (Lin et al., 2019). Others like RAT-SQL in Wang et al. (2020) use grammar generated from compiler tool which is

often redundant, opaque to understand and offering no flexibility in model design.

To address these challenges, we propose a robust, unified framework to solve Text-to-SQL problem. Inspired by a recent work (Herzig and Berant, 2020), the foundation of our parser is based on fine-grained query understanding. We leverage a span based named entity recognition (NER) model to chunk input query and extract SQL-typed entities. Based on the type information we link entities to database using neural entity linker (NEL) model. NEL provides linked literal values to the parser, thus the generated SQL is executable. The final module of our framework is a grammar-based seq2seq parser which synthesizes executable logical forms from natural language query (NLQ), schema, linking results and grammar. In our parser, we dynamically build logical form grammars from training data. This approach frees us from manually constructing grammars and streamlines development of parser model. At the same time, as grammar creation is agnostic to database, our framework is more general-purpose comparing with previous works. In addition to NLQ and schema, linking information from NEL is also fed to the parser to ensure global reasoning in the decoding. Concretely this linking feature will help guide model to select proper actions in decoding. A recent work (Ma et al., 2020) takes a similar approach as our framework — they have an extractor model to extract entities and then link mentions to database. However, they use entity label relations to construct logic forms which greatly limits complexity of generated program. By decoupling query understanding, linking and parsing, our framework offers better explainability and flexibility in model design and optimization.

A major challenge to build such pipeline system is fine-grained annotations which are needed to train entity recognition model and entity linking model. To tackle this issue, we leverage the newly released SQUALL dataset which provides alignment annotations between NLQ tokens and logical forms (Shi et al., 2020). Instead of using alignment signal as attention supervision as in Oren et al. (2020) and Shi et al. (2020), we programmatically convert the alignment annotations to entity annotations and linking annotations, and use these supervision signals to train NER and NEL models. A training example is shown in Figure 1 that illustrates how the aforementioned conversion works.

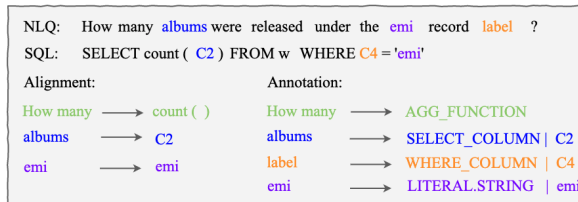


Figure 1: Fine-grained annotation from SQUALL dataset: based on target logical forms, alignment and database contents we can derive SQL-semantic annotation types for spans in query. Contents behind vertical bar in annotation are corresponding linking results.

We evaluate our framework on SQUALL dataset which has fine-grained alignment annotations. On the dev set, our framework obtains 49.36% logical form exact match (EM) accuracy and 69.14% execution accuracy, which is 2.2% and 2.6% improvement comparing with the best model in SQUALL paper, respectively.

2 Approach

In this section, we describe our pipeline framework and its application to chunking, linking and parsing tasks.

2.1 Problem Definition

The input to Text-to-SQL semantic parsing problem is a sequence of natural language query tokens $Q = \{q_1, q_2, \dots, q_{|Q|}\}$ and a relational database containing multiple tables $D = \{t_1, t_2, \dots, t_{|T|}\}$. Each table is represented as $T = \{h_1, h_2, \dots, h_{|H|}, c_1, c_2, \dots, c_{|C|}\}$ where h_i and c_i are column headers and cell values in a table, respectively. The goal of the task is to generate an output program Y consisting of a sequence of production rules from grammar, schema and literal values. In terms of structured decoding, an abstract syntax tree is generated and the best tree \hat{y} is computed by:

$$\hat{y} = \arg \max_y P(y|q, t, h, c)$$

given query tokens q and database contents including, table names t , column headers h and cell values c . Different from previous work (Guo et al., 2019), we are targeting at generating full-fledged SQL query which is directly executable.

2.2 Schema and Cell Value Aware NER Model

The first stage of our framework is an NER model which serves to understand user intents in the query.

Considering the fact that there could be nested entities, a span based NER model is used to chunk and identify entities in query (Eberts and Ulges, 2019; Zhong and Chen, 2020). We extract aggregation functions, column mentions and literal values from query. In addition, for columns we add SQL semantics to the NER tags. Specifically, as tags shown in Figure 1, we have fine-grained tags such as “WHERE_COLUMN”, “GROUPBY_COLUMN” etc. A pretrained BERT base model is used as its core (Devlin et al., 2019), as illustrated in Figure 2.

Unlike regular NER tasks, entities in this use case highly depend on underlying database contents. To this end, we design a schema and cell value aware NER model to take database information into account. As is shown in Figure 2, we append schema and cell values to query tokens as input to the BERT encoder and separate them using “[SEP]” token. Let $S = \{s_1, s_2, \dots, s_n\}$ denote all spans built from NLQ tokens. A span is represented as:

$$e_s = [e_{ctx}; e_{start}; e_{end}; e_{length}]$$

which is concatenation of representations of context e_{ctx} , start of span token e_{start} , end of span token e_{end} and learned span length embedding e_{length} . The span vector then goes through a multilayer perceptron to predict whether the span is an entity and determine the corresponding entity types. We minimize the negative log-likelihood for all spans during training.

$$p(y_s|q, t, h, c) = \text{softmax}(We_s + b)$$

$$L_{NER}(\theta) = - \sum \log p(y_s|q, t, h, c; \theta)$$

Here q, t, h, c have the same definition as Section 2.1 which represent query tokens, table names, table headers and cell values. θ are learnable parameters in NER model. As in Zhong and Chen (2020), a *None* token is added into vocabulary of entity types. At inference time, spans which are classified as *None* will be discarded.

As the first stage of our pipeline, recall of NER model has great impact on the system performance. To improve recall performance, we introduce an additional post-processing step where we collect all schema and cell values and use them as gazetteer list. When there are **exactly matched** spans in NLQ, we force model to generate a valid entity type for such spans. At the same time, if a span overlaps with a gazetteer matched span, we choose to keep gazetteer matched span as it is more likely to be a valid span.

To further leverage matching information, we add constrained decoding after filtering (Lester et al., 2020). Concretely in decoding process we force model to predict labels based on gazetteer matching category. For instance, when chunking query “*how many points does LeBron James get in last game*”, if there is an exact match of span “*LeBron James*” with an entry in cell value gazetteer list, the decoding logic will force the model to give a prediction tag which is compatible with cell value type.

2.3 Neural Entity Linking (NEL) Model

The sketch of generated SQL logical forms consists of rules from grammar. To populate the sketch we use a pointer network in parser to copy table names, column names and cell values from input schema and query. However, directly copying these entities will lead to non-executable SQL as columns and literal values in NLQ can be different from their canonical form in the database. Our entity linking model bridges the gap between entity mentions in query and entity values in SQL. Even though fuzzy string match is widely used for linking task in literature (Wang et al., 2020; Shi et al., 2020), in real application purely relying on fuzzy match could lead to suboptimal performance. For instance, in SQUALL dataset, there are less than 50% of entities that have exact matches in database. In light of this, we use a neural ranking model for entity linking task. Specifically, given a mention in NLQ and a list of candidates in database, NEL model selects the best matched candidate for each entity (Ledell Wu, 2020).

For column and literal value entities from output of NER model, we construct an input to the NEL model using NLQ tokens, mention and candidates. For example, input to NEL model can take the following format:

$$NLQ \text{ [SEP] } Mention \text{ [SEP] } Candidate$$

As we know the mention type from NER model, we could narrow down candidates to a specific type. For instance, if a mention is literal value type, then candidates are only limited to cell values. It is worth to note that without fine-grained query understanding, for each mention in NLQ, linking candidates have to be all contents in database. Thus it is challenging for end-to-end model to deal with cases where there are overlapping column names and cell values. In addition, we could append additional meta features to the input of NEL model.

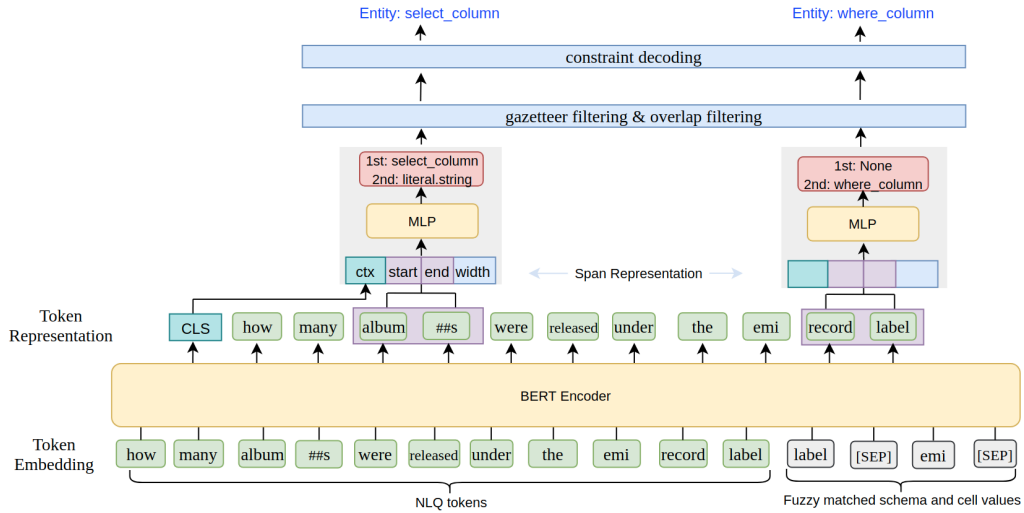


Figure 2: Span based schema and cell value aware NER model architecture. Input tokens are tokenized by BERT tokenizer.

When a mention is column type, we could construct the following input:

NLQ [SEP] *Mention* [SEP] *Candidate* [SEP] *value* [SEP] *column type*

where value is the cell value in current candidate column which has the highest fuzzy matching score with NLQ. Column type could be meta information such as data type *integer*, *string* etc.

In our experiments, we use BERT base model for linking model. After BERT encoding, a linear layer is applied on the classification token “[CLS]” to produce a logit score for each candidate. During training, cross-entropy loss is calculated over all the linking candidates. During inference, each linking candidate is fed to the BERT model independently and is scored by BERT model. Then best candidate is selected based on ranking scores.

2.4 Neural Semantic Parsing (NSP) Model

Neural semantic parser model takes as input NLQ, a database and outputs a sequence of production rules which can be used to deterministically build up an output program. The backbone of our neural semantic parser is a BART Large model which is pretrained with large amount of parallel text data for both its encoder and decoder (Lewis et al., 2020).

Encoder Encoder encodes NLQ tokens and schema information. Specifically, NLQ tokens, table names and column names are concatenated together with a “[SEP]” token used as separator. As we have already got literal value spans from

NER stage, at the output of BART encoder, we collect representations for all of these literal spans by pooling average token representations in the span.

From query understanding, we have SQL-semantic tags for each column mentions. For example, in Figure 1 we know that “album” is a “SELECT_COLUMN”. At the same time, in NEL results we know “album” is linked to column “C2”. Consequently, we know that “C2” is used in the logical forms as a selection column. In order to utilize this information in parser, we have a column type embedding layer in the encoder. When constructing column representations, we concatenate column type embeddings to the original column representations. To alleviate upstream errors, during parser model training, we randomly drop column type feature for 20% of time so that when NER model gives incorrect predictions, parser model learns to handle these cases. In our experiments, we will show that this feature can give big boost to parser performance.

Decoder The generated program at the output of a semantic parser can be a sequence of logical form tokens or an AST tree. The former decoding is generally referred to as flat decoding and the latter one is called structured decoding. To synthesize syntactically correct SQL program, in our framework a grammar based autoregressive top-down decoder is utilized to generate AST. Contrary to implementation in Yin and Neubig (2017) where AST grammar is collected through compiler’s tool, we dynamically generate context free grammar from training data. Concretely, during training stage, we

collect ground truth SQL queries and parse them into SQL trees. Then we collect all the rules as our grammar using breadth-first search algorithm. The distinguishing feature of our grammar generation comparing with the one discussed in Lin et al. (2019) is that we don't need so-called "linked rules" because we get linked entity from NEL results. This method saves us from manually writing rules for each dataset. At the same time, it decouples SQL grammar from domain knowledge which makes our framework more general-purpose than previous works.

At each decoding step, decoder takes as input previous decoding result and iteratively apply production rules to non-terminal nodes. Owing to our query understanding based framework, we are able to employ soft copy mechanism (See et al., 2017) to directly copy tables, columns and values from output representations of encoder. Finally, beam search is used during inference time.

3 Experiment

3.1 Data

SQUALL (Shi et al., 2020) is collected based on WikiTableQuestions which is a question-answering dataset over structured tables. In SQUALL, each query only relies on one table to get the answer.

To obtain supervision labels for NER and NEL model, we first parse ground truth SQL queries into trees. Then based on alignment information provided by the dataset, we programmatically derive entity labels and linking labels for each entity span in the query. In total, there are 11276 training instances and 4344 testing instances in the dataset. Train, dev, test set partition is based on the pre-defined setting in the released dataset. We use logical form exact match accuracy and execution accuracy as our evaluation metric.

3.2 Model Analysis

We first explore the best setup for our framework. In these experiments, we always use structured decoding in the parser. Based on how to utilize query understanding results, there are three different configurations of our parser model: (1) In our baseline model setup, we only utilize NER and NEL results for entity linking purpose and copy linked results into the generated AST. (2) Instead of using all schema in parser's encoder, we only input linked columns to the parser (3) We inject fine-grained query understanding results in the parser, i.e. add

Model	Dev	
	ACC _{LF}	ACC _{EXE}
(1) Baseline	42.56	60.57
(2) Linked columns only	38.69	56.83
(3) Columns type feature	49.38	69.14
(4) Oracle column type feature	68.21	85.16

Table 1: System performance with different approaches to utilize query understanding results in semantic parser. ACC_{LF}, ACC_{EXE} are logical form accuracy and execution accuracy, respectively.

linked column type information as meta features to the encoder of NSP.

In Table 1, we summarize performance of our system on dev set under different parser configurations. Comparing "(2) Linked columns only" model with baseline model (1), we can see that system performance suffers because when we only use linked columns in the parser, NER model errors will propagate to the downstream. With adding column type feature based approach (row 3 in Table 1), we find that it can greatly boost model performance as it guides parser to choose correct columns. We also have an oracle experiment where we use ground truth column type feature. As is shown in last row in Table 1, it's around 19% better than our best configuration (row 3 in Table 1) which means there is still room to improve our system.

In Table 2, we compare our model performance with the best model (ALIGN) in SQUALL paper. Their end-to-end model leverages BERT as the encoder and LSTM as decoder using flat decoding strategy. They're using supervised attention to help model learn alignment information. In order to have a fair comparison with SQUALL paper, we add two more baselines in our experiments: in the first experiment we augment ALIGN model by replacing BERT encoder and LSTM decoder with BART model; in the second experiment, we replace structured decoding with flat decoding in our system. In a nutshell, we compared performances of four models: (1) original ALIGN model from Shi et al. (2020), (2) our augmented implementation of ALIGN, (3) our framework with flat decoding, (4) our best configuration (row 3 in Table 1). As can be seen from the table, our best model outperforms ALIGN model and augmented ALIGN model in both logical form accuracy and execution accuracy. On test set we achieve 56.8% execution accuracy which is 2.7% higher than the ALIGN model. If

Model	Dev		Test
	ACC _{LF}	ACC _{EXE}	ACC _{EXE}
(1) ALIGN(SQUALL)	47.2	66.5	54.1
(2) ALIGN(SQUALL) + BART	47.7	67.1	54.6
(3) Ours + Flat decoding	49.1	68.8	56.2
(4) Ours	49.4	69.1	56.8

Table 2: Performance comparison of our model with ALIGN model in SQUALL paper. ALIGN + BART model is our implementation of ALIGN model where we replace BERT encoder and LSTM decoder with BART encoder and decoder.

Model	ACC _{LF}
ALIGN(SQUALL)	30.29
ALIGN(SQUALL) + BART	30.94
Ours	34.78

Table 3: Model’s performance on nested queries. We retrained ALIGN model for this experiment.

we compare two ALIGN models and two of our systems with different decoding strategy, it’s easy to tell BART model and structured decoding contribute limited benefits in our framework which in turn suggests that major improvement in our system is from fine-grained query understanding part.

We also evaluate the generalizability of our model. Our assumption is that span based NER model can chunk query based on how meaning is composed. As we add NER label information into parser, we hypothesize that the parser can learn to generate program based on semantic type of query tokens rather than just using lexical meaning of tokens. Thus, we would see better performance on compositional generalization. Particularly, we want to see how model works on nested queries since it is an ideal set to evaluate model’s ability on generalization. To this end, we collect all nested queries in dev set and evaluate our system on this nested query set. The performance is shown in Table 3. Our model shows 4.5% improvement comparing with original ALIGN model and around 3% improvement comparing with augmented ALIGN model. The results demonstrates that our framework is better at compositional queries.

3.3 Ablation Study

As the first module of our framework, NER model plays a critical role in our pipeline system. To qualitatively study the impact of NER model, we did ablation experiments to study how each component in

Model	Dev Set	
	NER F1	System ACC _{LF}
Our best model	85.14	49.38
-cell	84.23	47.51
-schema	83.70	47.66
-gazetteer	84.15	45.89
-gazetteer-cell-schema	82.98	43.66

Table 4: NER performance ablation study. Baseline model here is span-based schema and cell value aware NER model. We gradually remove each component to see its impact on the system performance.

NER model affects the system performance. Concretely, we use our schema aware and cell value aware NER model as the baseline and gradually remove each component to see how system performance fluctuates. Table 4 summarizes our findings. As is shown in the table, when we remove cell values, schema and gazetteer filtering, NER F1 score goes down and system performance degrades accordingly. We can also see that among these three components, system performance drops the most (from 49.38 to 45.89) when we remove gazetteer filtering. Gazetteer filtering in NER serves the role to combine string match and model prediction. It forces NER model to output predictions for exactly matched spans at output which increases recall of NER model. From system perspective, downstream parser is more sensitive to missing entities. Thus, improving NER recall can greatly boost system performance.

4 Conclusion

In this work, we proposed a novel, general-purpose Text-to-SQL semantic parsing framework which is based on fine-grained query understanding. The framework tackles several pain points in the Text-to-SQL problem and offers a new robust approach for real-life applications. Our framework outperforms previous state-of-the-art result by 2.7% on SQUALL test set. In the future, we plan to explore using fine-grained query understanding results to constrain decoding search space in parser to further improve system performance.

5 Limitations

While this work aims to improve Text-to-SQL semantic parsing with fine-grained annotations, we don’t have enough time and resources to collect a dataset for such purpose. Due to this issue, our

experiments are limited to SQUALL dataset. In the future, we plan to build a comprehensive dataset to facilitate research in the area.

References

- Ursin Brunner and Kurt Stockinger. 2020. Valuenet: A neural text-to-sql architecture incorporating values. *ArXiv*, abs/2006.00888.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Markus Eberts and Adrian Ulges. 2019. [Span-based joint entity and relation extraction with transformer pre-training](#).
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.
- Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.
- Martin Josifoski Sebastian Riedel Luke Zettlemoyer Ledell Wu, Fabio Petroni. 2020. Zero-shot entity linking with dense entity retrieval. In *EMNLP*.
- Brian Lester, Daniel Pressel, Amy Hemmeter, Sagnik Ray Choudhury, and Srinivas Bangalore. 2020. [Constrained decoding for computationally efficient named entity recognition taggers](#). In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1841–1848, Online. Association for Computational Linguistics.
- 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.
- Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *ArXiv*, abs/1905.13326.
- Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. [Mention extraction and linking for SQL query generation](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6936–6942, Online. Association for Computational Linguistics.
- Inbar Oren, Jonathan Herzig, Nitish Gupta, Matt Gardner, and Jonathan Berant. 2020. Improving compositional generalization in semantic parsing. *arXiv preprint arXiv:2010.05647*.
- Ohad Rubin and Jonathan Berant. 2020. Smbop: Semi-autoregressive bottom-up semantic parsing. *ArXiv*, abs/2010.12412.
- Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard - parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- A. See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *ACL*.
- Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to SQL queries. In *Findings of EMNLP*.
- Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.
- Pengcheng Yin and Graham Neubig. 2017. [A syntactic neural model for general-purpose code generation](#). In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.
- Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. 2020. [Photon: A robust cross-domain text-to-SQL system](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 204–214, Online. Association for Computational Linguistics.
- Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for joint entity and relation extraction. *arXiv preprint arXiv:2010.12812*.

Unsupervised Dense Retrieval for Scientific Articles

Dan Li and Vikrant Yadav and Zubair Afzal and Georgios Tsatsaronis

Elsevier

{d.li, v.yadav, zubair.afzal, g.tsatsaronis}@elsevier.com

Abstract

In this work, we build a dense retrieval based semantic search engine on scientific articles from Elsevier. The major challenge is that there is no labeled data for training and testing. We apply a state-of-the-art unsupervised dense retrieval model called Generative Pseudo Labeling that generates high-quality pseudo training labels. Furthermore, since the articles are unbalanced across different domains, we select passages from multiple domains to form balanced training data. For the evaluation, we create two test sets: one manually annotated and one automatically created from the meta-information of our data. We compare the semantic search engine with the currently deployed lexical search engine on the two test sets. The results of the experiment show that the semantic search engine trained with pseudo training labels can significantly improve search performance.

1 Introduction

Search engines are deeply integrated into Elsevier’s information services of its scientific literature data. An example is the one provided by ScienceDirect¹, providing researchers with search services on more than 19M full text articles. These search engines are currently based on lexical search models such as BM25. The deployment of such models is effortlessly simplified by using popular industry-standard libraries such as Elasticsearch². However, lexical search suffers from the lexical gap problem such as misspellings, synonyms, abbreviations, ambiguous words, and ignoring of word order (Formal et al., 2021).

Recently, dense retrieval (DR) models have proven to be highly effective in solving the lexical gap problem while still remain fast search speed (Karpukhin et al., 2020; Xiong et al., 2020). DR models map queries and passages to a common vector space and retrieve relevant passages

¹<https://www.sciencedirect.com>

²<https://www.elastic.co>

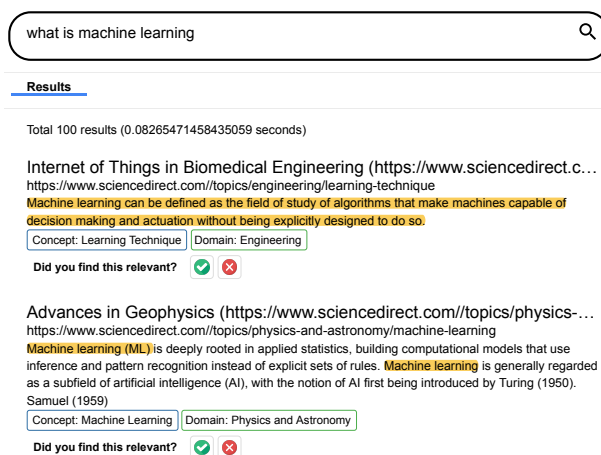


Figure 1: Interface of our semantic search engine.

by searching for (approximate) nearest neighbors. DR has been well studied on laboratory data but still in the early stage for industry-level applications (Hofstätter et al., 2022; Kim, 2022). DR is mainly applied in multi-modal search in industry where traditional lexical search is not possible, like text-image search (Radford et al., 2021) or music search (Castellon et al., 2021).

It is of great interest to use state-of-the-art DR models to build semantic search engines for industry. Such search engines can enable efficient access and search to scientific literature of Elsevier and help researchers in their journey (Elsevier, 2022). Our goal in this work is to develop a semantic search engine that needs no relevance-labeled data to train the DR model, thus allowing easy adaptation to new domains and easy deployment in industry.

There are several challenges to be tackled. First, training a DR model requires sufficient labeled data such as MS-MARCO (Nguyen et al., 2016), whereas there is often no such data for specific domains or startups. In our case, we have a large collection of passages from scientific articles but no relevance label. Furthermore, it is shown that DR

models trained on one domain do not generalize to another (Thakur et al., 2021). The passages in our corpus have a different word distribution compared to that in MS-MARCO. Besides, the passages are also unbalanced regarding their domains (see Section 4.3). Therefore, using the models trained on MS-MARCO will not yield high retrieval performance. It is interesting to tackle the domain difference problem. Finally, there is no test set to evaluate search performance and creating a good-quality test set is time-consuming and expensive. All these challenges hinder the application of DR models in industry setting.

In this work, we trained a DR model using a state-of-the-art unsupervised dense retrieval model called GPL (Wang et al., 2021). It uses a pre-trained query generator to generate queries from passages. The passage is considered as positive for the generated queries. Negative passages for generated queries are retrieved using existing dense retrieval models trained on MS-MARCO. An existing cross-encoder model trained on MS-MARCO produces relevance scores of query-passage pairs as supervision signals to train the DR model.

Finally, we constructed two test sets by either manual annotation or automatic extraction of existing relevance information from the meta field of the corpus. The experimental results show that our best model can significantly improve the retrieval performance compared to lexical and semantic search baselines.

The semantic search engine we have created for our product is shown in Figure 1. It is currently deployed and running in a beta test mode.

2 Related Work

2.1 Dense retrieval

The very first work on dense retrieval (DR) was proposed by Karpukhin et al. (2020). DR uses text encoders to represent queries and documents as dense vectors and retrieve documents by similarity scores between query vectors and document vectors. It has shown to achieve competitive performance in first-stage retrieval compared with traditional lexical retrieval method.

Researchers have been working towards improving the effectiveness of DR models through negative sampling (Xiong et al., 2020; Zhan et al., 2021; Lin et al., 2021), pre-trained language models (Gao and Callan, 2021), and pseudo relevance labels (Prakash et al., 2021; Yu et al., 2021), as well

as improving the efficiency of DR models with sparse representation (Zhan et al., 2022; Thakur et al., 2022).

2.2 Unsupervised dense retrieval

Unsupervised dense retrieval (UDR) aims to train dense retrieval models without manually labeled data. It generates high-quality pseudo labeled data and designs proper loss functions to train DR models.

The first step is to generate positive examples, which is done by extraction or generation. For example, Izacard et al. (2021) extracted a pair of relevant texts from the same document using the inverse cloze task and independent cropping. Wang et al. (2021) generated queries from documents using existing encoder-decoder as positive examples.

The second step is to generate negative examples. Izacard et al. (2021) used contrastive loss to create negative batches within a batch and across batches. Wang et al. (2021) used existing weak retrievers to retrieve top-k documents as negatives.

The third step is to design training loss. Due to the noisy fact of pseudo examples, traditional pairwise ranking loss (Burgess, 2010) is not a good choice because the training are easily affected by noisy labels. Instead, contrastive loss is widely used (Izacard et al., 2021; Xu et al., 2022). On the other hand, relevance scores from existing generalizable cross-encoder have been used as supervision signal (Wang et al., 2021).

3 Methodology

3.1 GPL Model Training

Since there are no relevance labels for the passages in our corpus, we apply a recent unsupervised dense retrieval model GPL (Wang et al., 2021) to train our dense retrieval model. We generate 3 queries from each passage using a pre-trained query generator (Nogueira et al., 2019). The passage-query pairs will be the pseudo positive examples. For each generated query, we retrieve similar passages using two existing DR models trained on the MS-MARCO dataset (Reimers and Gurevych, 2019), and take the first 50 of each model as pseudo negatives. Finally, we use a student-teacher training method. The teacher model is a cross-encoder trained on MS-MARCO which shows good performance in zero-shot retrieval tasks (Hofstätter et al., 2020). The student model is the bi-encoder DR model to be learned.

The student-teacher training is used because the pseudo labels are noisy and can not be directly used in the traditional pairwise ranking loss (Burgess, 2010) or contrastive loss (Wu et al., 2018). Instead, using a cross-encoder has been demonstrated to generalize well on different datasets (Hofstätter et al., 2020) and thus can be used as a teacher model through knowledge distillation.

For the knowledge distillation we have used MarginMSE loss (Hofstätter et al., 2020). It is defined as:

$$L_{MarginMSE} = -\frac{1}{M} \sum_{i=0}^{M-1} |\hat{\delta}_i - \delta_i|^2 \quad (1)$$

$$\hat{\delta}_i = f_{be}(q_i)^T f_{be}(p_i^+) - f_{be}(q_i)^T f_{be}(p_i^-)$$

$$\delta_i = f_{ce}(q_i, p_i^+) - f_{ce}(q_i, p_i^-),$$

where f_{be} is the bi-encoder, which maps the text of query or passage to a vector, f_{ce} is the cross-encoder, which maps the text of query and passage to a score, q_i is the query, p_i^+ is the positive passage, and p_i^- is the negative passage.

By minimizing $L_{MarginMSE}$, the MarginMSE loss avoids the hard treatment of the positives and negatives as in pairwise ranking loss (Burgess, 2010) and contrastive loss (Wu et al., 2018). For example, for (*false positive, negative*) pairs or (*positive, false negative*) pair, we do not expect the bi-encoder to put them far away in the embedding space or have small similarity scores. The cross-encoder will assign a small δ value and the MarginMSE loss will teach the bi-encoder to produce small $\hat{\delta}$ value as well.

Implementation. We use *all-t5-base-v1* as the query generator because it is designed to generate key-word queries, which is similar with the terms or topics people search in our product. We use *msmarco-distilbert-base-v3* and *msmarco-MiniLM-L-6-v3* as the negative retrievers, and *ms-marco-MiniLM-L-6-v2* as the teacher cross-encoder as suggested in GPL. We use *sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco* as the starting checkpoint of the student bi-encoder because this is the best bi-encoder on MS-MARCO. The teacher model and the student model contain 22M and 66M parameters, respectively. All the models aforementioned can be downloaded from Huggingface³. We set batch size 16. We set maximum sequence length 512. Note that the passages are snippet from the articles and have on average

³<https://huggingface.co/models>

474 English words or 723 WordPiece (Wu et al., 2016) tokens. Cutting off of the passages loses information. It is worthy split the passages into shorter ones and we leave the work for future study.

3.2 Test Set Construction

Corpus The corpus we are working on supports a web service providing concept definitions and subject overviews for researchers who want to expand their knowledge about scholarly and technical terms.⁴ For example, for the term “water purification”, a web page is created that contains its definition, several scientific article snippets containing other definitions of the term, and several relevant terms as well. The corpus contains about 2M passages extracted from scientific articles. The articles are from 20 different domains and not evenly distributed across domains. Figure 3 shows the domain distribution.

Manual test set. We aim to develop a semantic search engine on top of this corpus, so that when a user searches a term, the semantic search engine returns passages containing the definition of the term. Therefore, the ideal queries are questions about scientific terms, and the ideal relevant passages are those talks about (part of) the definitions the terms.

As the data contains scientific terms from 20 domains, we sample one term from each. We only sample those having Wikipedia pages to increase the chance that there exists relevant passages for a query.

We use the widely-used pooling method in information retrieval (Ferro and Peters, 2019) to select passages for annotation. We include 3 different retrieval systems in the pool including the BM25 model (Pérez-Iglesias et al., 2009), the TAS model (Hofstätter et al., 2021), and the GPL model trained by us, in order to ensure the passages in the test set are diverse and not biased towards either lexical retrieval or semantic retrieval methods. We randomly sample from the top-10 passages in the ranking lists.

We had 3 workers annotating the selected query-passage pairs. Conflicts of annotation were discussed until an agreement was reached. Finally, 20 queries and 539 query-passage pairs are selected and annotated.

Auto test set. Although the manual test set has high quality, it is too small and thus sensitive for

⁴<https://www.sciencedirect.com/topics/index>

evaluation. We use the noisy meta information in our corpus and construct a larger test set. The passages in the corpus is organised by terms. Each term has several passages associated with it that are considered relevant and containing the definition of the term. The extraction of the definition and relevant passages are done by production system based on lexical methods. Thus the passages can be roughly taken as relevant to that term. To balance terms from different domains, we sample 10 terms from each domain and take all the passages associated with it as relevant. Finally, we have 200 queries and 3,562 relevant labels.

3.3 System Architecture

Figure 2 shows the architecture of the semantic search engine. The system is divided into two parts, offline and online. In the offline part, we download the corpus from Amazon S3 buckets, then on Amazon Sagemaker we preprocess the corpus, train the the bi-encoder model and convert the passages into 768-dimensional vectors using the trained model. The HNSW⁵ algorithm is used to index the passages.

The online part is divided into two parts. One of the parts is an API-based service running on Amazon Sagemaker. The task of this service is to convert the user query into a vector and find the passages closest to the query vector using the index we created in offline mode. The other part is a UI based interface running on an Amazon EC2 instance. This part processes the user query and displays the passage associated with the query through a UI interface.

The EC2 instance and API run on an Intel Xeon-based processor and the cost of running them is 1 dollar per hour. For training the model, we use the AWS p3.8x.large EC2 instance type. This instance is installed with NVIDIA Tesla V100 GPUs. The cost of training the model was approximately 200 dollars. During inference time, the system is running on a CPU instance and it is able to process up to 70 requests/second. The average time needed to get the search result for a query is 0.03 second.

4 Experimental Setup

4.1 Research Questions

- (RQ1) How does the model perform compared with the current production model and other baselines?

⁵<https://github.com/nmslib/hnswlib>

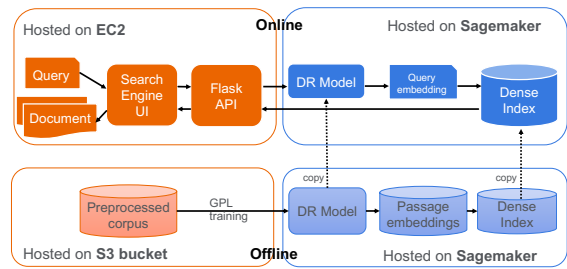


Figure 2: Architecture of the semantic search engine.

- (RQ2) Is it necessary to use the whole corpus to train the model?
- (RQ3) Whether balancing passages from different domains in training batches improves model performance?

4.2 Baselines

BM25. This baseline is the current search engine in production. It uses lexical retrieval model BM25 implemented in Elasticsearch.

TAS (Hofstätter et al., 2021). The *sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco* model is a zero-shot baseline. It was the best bi-encoder on MSMARCO when this paper was submitted. We also use this model as the starting checkpoint to train the GPL model.

BM25+CE. This is a two-stage baseline implemented by us. It consists of lexical retrieval and re-ranking. We first use BM25 to produce a ranked list of passages, then use a cross-encoder *ms-marco-MiniLM-L-6-v2* trained on MSMARCO to rerank the top-1000 passages. We use this model as the teacher model when training GPL.

4.3 Dataset

We use two test sets including the Manual and the Auto. Table 2 shows the statistics. The Manual has 20 queries and the Auto has 200 queries. Auto also has more labels for query-passage pairs. Note there is 0 non-relevant labels for Auto, however this does not affect the evaluation as all the rest passages without a relevant label will be counted as non-relevant. To speed up evaluation, we randomly sample a subset of passages for the models to retrieve from, combined with the passages in each of the two test sets. This results in two test corpora consisting of 100,513 and 102,506 passages for the Manual and the Auto. The test corpora have the same domain distribution with the full corpus.

Model	Manual test set				Auto test set			
	NDCG@10	MAP@10	MRR@10	R@100	NDCG@10	MAP@10	MRR@10	R@100
BM25 ¹	61.86	42.59	77.38	87.05	53.08	24.17	68.81	70.81
TAS	47.78	25.98	74.17	67.64	28.20	9.97	46.81	40.16
GPL	71.29	42.42	85.70	91.71	49.78	22.44	74.21	59.41
GPL_BLC	74.42	44.96	87.62	91.77	50.16	22.47	75.49	59.69
BM25+CE ²	84.90	54.96	95.00	90.99	68.33	36.87	86.68	78.56

¹ Production model of our product.

² Upper bound of our model.

Table 1: Retrieval performance (%). The best values for each metric and the upper bound method is in bold.

	Manual	Auto
Test set		
# query	20	200
# passage	539	2614
# relevant	289	3562
# non-relevant	251	0
Test corpus		
# passage	100513	102506

Table 2: Statistics of test sets.

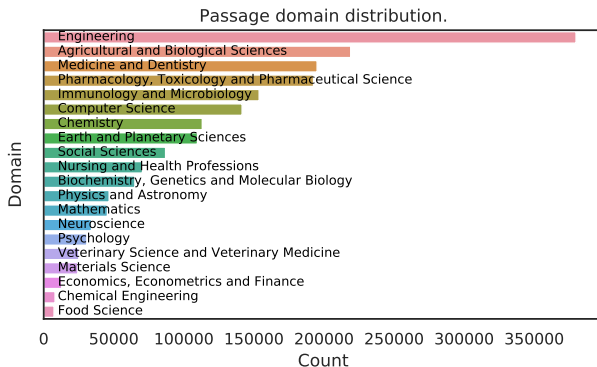


Figure 3: Passage domain distribution. The top 5 domains cover about 58.1% of the passages and the bottom 5 domains only contains about 3.97% of the passages.

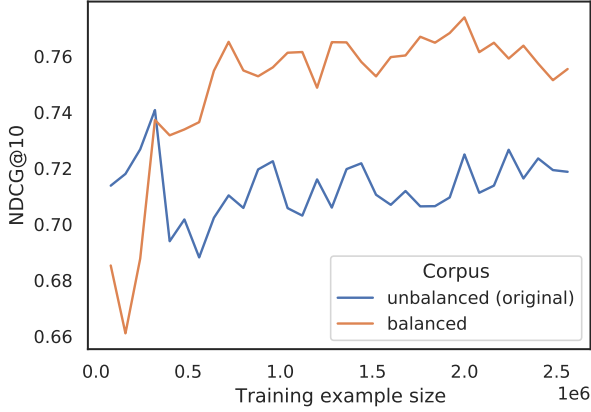
5 Results

5.1 Retrieval performance

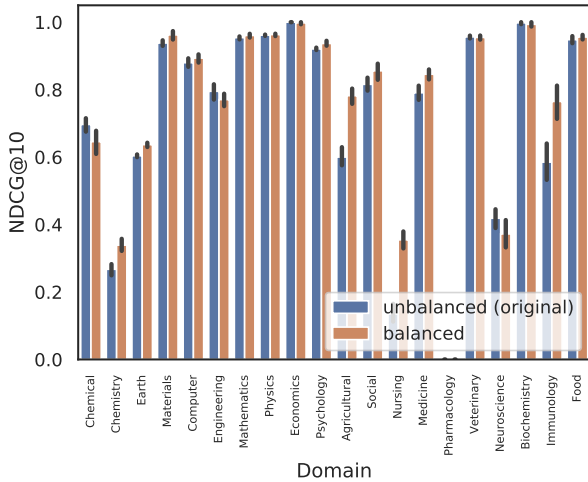
In this section, we aim to answer RQ 1. We use a subset of 83K passages from our corpus and generate 3 queries for each passages and generate 100 negative passages for each query. Finally, we sample 4M training examples in the format of $(q_i, p_i^+, p_i^-, \delta_i)$. It is suggested that such a volume is enough to train a GPL model for a new domain (Wang et al., 2021). We also empirically demonstrate the impact of training example size in Section 5.2. We train two GPL models: GPL is trained on 83K passages randomly sam-

pled. GPL_BLC is trained on 83K passages which are balanced sampled from the 20 domains. Since we aim to build a one-stage retrieval model, we compare our model with a lexical retrieval model – BM25 and a zero-shot dense retrieval model – TAS. We also compare with a two-stage method – BM25+CE.

Table 1 shows the retrieval performance. First, BM25 performs robustly well on the two test sets, while zero-shot TAS performs poorly. It indicates that dense retrieval models do not generalize well on new domain. This finding is consistent with the work of Thakur et al. (2021). The difference of metrics between BM25 and TAS is larger on Auto, because we have annotated both lexical and semantic relevant passages in the Manual test set while most relevant passages in the Auto test set are obtained by lexical methods only. The dense retrieval model TAS is thus down-estimated on Auto. Second, BM25+CE performs the best. It improves NDCG@10, MAP@10, and MRR@10 to a large margin compared to BM25. The cross-encoder model (*ms-marco-MiniLM-L-6-v2*) is trained on MS-MARCO. Thus, the result indicates the good generalization capability of cross-encoder ranking models. Third, GPL or GPL_BLC perform better than BM25 on most the metrics and better than TAS on all the metrics. For example, an MRR@10 of 87.62 means that GPL_BLC can rank relevant passages on the first or second position on averaged queries, an R@100 of 91.77 means that GPL_BLC can retrieve 91.77% of the relevance passages in top 100. Note that the performance difference between GPL and GPL_BLC is big on Manual but small on Auto. The possible reason is that on Auto most semantically relevant passages are not labeled in the test set.



(a) NDCG@10 averaged over queries against training example size.



(b) Query-wise NDCG@10 of model trained with all the 4M training examples.

Figure 4: NDCG@10 of the Unbalanced and Balanced corpus.

5.2 The impact of training example size

In this section, we aim to answer RQ 2. We use all the 2M passages in the corpus and generate 32M training examples to train the model. We save the checkpoint every 160K examples. We evaluate model performance on the Manual test set. Figure 5 shows the NDCG@10 score against the training example size. We observe that more training examples do help to improve the performance of the model. The performance increases fast at the beginning and achieves an NDCG@10 of 0.74 with about 1M training examples, it then increases slowly towards an NDCG@10 of 0.80.

To sum up, it is not necessary to train the GPL model with all passages in our corpus; a volume of 1M training examples should be sufficient for the model.



Figure 5: NDCG@10 of GPL model trained with different number of examples. The x -axis is from 0.1×10^7 to 3.2×10^7 .

5.3 The impact of domain distribution

In this section, we aim to answer RQ 3. Since there is meta information about what domain the passages belong to in our corpus, we compare the model trained on the random 83K passages (*Unbalanced*) and the model trained on the 83K evenly distributed in the 20 domains (*Balanced*). Figure 4 shows the NDCG@10 of corpus 83K and 83K-balance. We use the Manual set as the test set. We observe that (1) there is a large improvement on 83K-balanced compared to 83K-unbalanced; (2) the NDCG@10 increase for most queries, and the improvement is especially large for those with low NDCG@10.

5.4 Case study

In this section, we show one query and the top 3 ranked passages selected from the Manual test set to analyze the retrieval effectiveness. We showcase three models including BM25, TAS, and GPL. The case study helps us to know how the retrieved passages are different for the DR model trained on the target domain, the zero-shot DR model and the lexical retrieval model. BM25, as expected, retrieves passages containing exact match of words in the query. As it is a bag-of-words model, we observe that the word “water” and “purification” do not always appear together in the passages. TAS can retrieve semantically similar passages, but they are sometimes off the topic. For example, the 1st passage retrieved by TAS is about “fuel purification”, it even contains the definition. However, it is not about “water purification”. TAS_GPL can retrieve relevant passages which even contain the definition.

Model	BM25	TAS	GPL
Query	What is Water Purification		
1st passage	...Importance of purification. Physicochemical properties of our model system. Adsorption layer of a nonionic surfactant. Ionic surfactant at the air water interface...	...The purification process is shown schematically in Figure 7-38. Fuel purification is a one-stage extraction procedure which employs centrifuges to treat distillates...	...Water purification for human consumption purposes consists in the removal of different contaminants as chemicals (i.e., pollutants, toxic metals), biological contaminants...
Relevance	0	0	2
2nd passage	... Such basic issues have to be addressed ahead of any assessment of water purification technologies, since such purification may not even be necessary...	...Purification is practical with distillate fuel and light crude oils having a minimum 0.5% water in the fuel, with a...	...Fuel purification is a one-stage extraction procedure which employs centrifuges to treat distillates and light crude oils without adding water...
Relevance	1	0	0
3rd passage	...Preparation of clarified growth media from an overnight culture of bacterial cells is the first and perhaps most important step in purifying OMVs. Before proceeding to purification...	...Disinfection, the desired result of field water treatment, means the removal or destruction of harmful microorganisms. Technically, it refers to chemical means such as...	The terms "water treatment" and "water purification" are extensively used for any unit operations and processes that involve methods and processing steps ...
Relevance	0	2	2

Table 3: Case study.

For example, the 1st passage is a good definition of "water purification". The case clearly shows that lexical retrieval and dense retrieval find very different passages. This is because their ways of representing texts are completely different. Furthermore, training DR models on the target domain can improve retrieval performance to a large margin even though the training labels are noisy.

6 Conclusions & Future Work

In this work, we build a semantic search engine on scientific articles. To tackle the challenge of no labeled data for both training and test, we apply a state-of-the-art unsupervised dense retrieval model named GPL. As the articles are unbalanced across different domains, we sample passages from multiple domains to form balanced training batches. We also created two test sets for the evaluation: one manually annotated and one automatically constructed from the meta information of our corpus.

We compare the semantic search engine with the currently deployed lexical search engine on the test sets. Both the qualitative and quantitative experiment results show that the semantic search engine can significantly improve the search performance. This results suggest that GPL is a robust and effective model for unsupervised dense retrieval.

For the future work, we will train the query generator and the negative retriever on our data to generate a better quality of both positive and negative

training example to improve the retrieval performance.

7 Limitations

Currently, we see 3 limitations in our work. First, the query generator is trained on a different domain, which results in skipping important keywords or phrases around which the query should be generated. Second, the negative retrievers are not adapted to the domain. The results obtained by these retrievers are negative but not "hard negative". This leads to limitations in learning of the student model. Third, the semantic search engine we build has not been evaluated on production population. We plan to conduct online evaluation in the future.

References

- Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.
- Rodrigo Castellon, Chris Donahue, and Percy Liang. 2021. Codified audio language modeling learns useful representations for music information retrieval. *arXiv preprint arXiv:2107.05677*.
- Elsevier. 2022. [All elsevier digital solutions](#).
- Nicola Ferro and Carol Peters. 2019. *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*, volume 41. Springer.
- Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292.
- Luyu Gao and Jamie Callan. 2021. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253*.
- Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *arXiv preprint arXiv:2010.02666*.
- Sebastian Hofstätter, Nick Craswell, Bhaskar Mitra, Hamed Zamani, and Allan Hanbury. 2022. Are we there yet? a decision framework for replacing term based retrieval with dense retrieval systems. *arXiv preprint arXiv:2206.12993*.
- Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. In *Proc. of SIGIR*.
- Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Towards unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.
- Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.
- Yubin Kim. 2022. Applications and future of dense retrieval in industry. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3373–3374.
- Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-batch negatives for knowledge distillation with tightly-coupled teachers for dense retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 163–173.
- 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 *CoCo@ NIPS*.
- Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019. From doc2query to docttttquery. *Online preprint*, 6.
- Joaquín Pérez-Iglesias, José R Pérez-Agüera, Víctor Fresno, and Yuval Z Feinsein. 2009. Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*.
- Prafull Prakash, Julian Killingback, and Hamed Zamani. 2021. Learning robust dense retrieval models from incomplete relevance labels. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1728–1732.
- 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. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.
- 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*. Association for Computational Linguistics.
- Nandan Thakur, Nils Reimers, and Jimmy Lin. 2022. Domain adaptation for memory-efficient dense retrieval. *arXiv preprint arXiv:2205.11498*.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.
- Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval. *arXiv preprint arXiv:2112.07577*.
- 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*.
- Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings*

of the *IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval. *arXiv preprint arXiv:2203.06169*.

HongChien Yu, Chenyan Xiong, and Jamie Callan. 2021. Improving query representations for dense retrieval with pseudo relevance feedback. *arXiv preprint arXiv:2108.13454*.

Jingtao Zhan, Jiabin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1503–1512.

Jingtao Zhan, Jiabin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2022. Learning discrete representations via constrained clustering for effective and efficient dense retrieval. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1328–1336.

Learning Geolocations for Cold-Start and Hard-to-Resolve Addresses via Deep Metric Learning

Govind
Amazon
gvindmg@amazon.com

Saurabh Sohoney
Amazon
sohoney@amazon.com

Abstract

With evergrowing digital adoption in the society and increasing demand for businesses to deliver to customers doorstep, the last mile hop of transportation planning poses unique challenges in emerging geographies with unstructured addresses. One of the crucial inputs to facilitate effective planning is the task of geolocating customer addresses. Existing systems operate by aggregating historical delivery locations or by resolving/matching addresses to known buildings and campuses to vend a high-precision geolocation. However, by design they fail to cater to a significant fraction of addresses which are new in the system and have inaccurate or missing building level information. We propose a framework to resolve these addresses (referred to as hard-to-resolve henceforth) to a shallower granularity termed as neighbourhood. Specifically, we propose a weakly supervised deep metric learning model to encode the geospatial semantics in address embeddings. We present empirical evaluation on India (IN) and the United Arab Emirates (UAE) hard-to-resolve addresses to show significant improvements in learning geolocations i.e., 22% (IN) & 55% (UAE) reduction in delivery defects (where learnt geocode is $>Y$ meters¹ away from actual location), and 43% (IN) & 90% (UAE) reduction in 50th percentile (p50) distance between learnt and actual delivery locations over the existing production system.

1 Introduction and Motivation

Last Mile delivery planning systems aim to optimize the delivery experience for both customers and delivery associates when packages travel from the final delivery stations to customer doorsteps. One crucial input to this planning is the delivery location of customers. Customers provide information regarding their whereabouts through address text, the only mandatory input they need

¹In this paper, the exact values at few places are not revealed due to the business confidentiality reasons and finer address details are masked (X) to preserve customers' privacy.

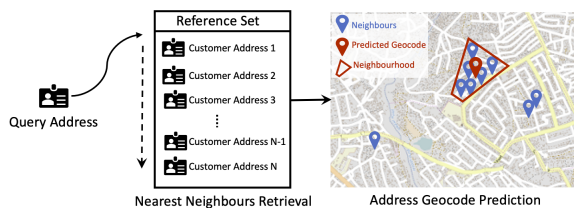


Figure 1: Address geocoding via nearest neighbours

to provide while placing their order. The task of learning geolocation of addresses is commonly known as geocoding and it is challenging in emerging geographies because of two primary reasons – 1) Lack of standardisation in the address text in form of spelling variations, missing components and use of vernacular content, synonyms and abbreviations, 2) Large proportion of cold-start addresses to which too few or no deliveries have been made in the past. For instance, the address *Bank Colony Sheriguda, Ibrahimpatnam, Gandhi Statue, 501510, Hyderabad, IN* does not contain any fine-grained details other than mentions of locality and landmark. Apart from the unstructured nature, addresses in emerging geographies tend to have inaccurate components such as *XX¹ Marina Bay One, Rawdat Al Reef, Abu Dhabi, UAE* contains wrong customer chosen district information (correct: *Al Reem Island*). It should be noted that the geocoding problem becomes trivial and simply reduces to aggregation of past delivery scans once there are successful deliveries to the address, irrespective of the address quality. The central theme of this work is to deal with customer addresses which have little or no delivery history along with missing or inaccurate components, also referred to as hard-to-resolve addresses here.

In general, the address geocoding task is largely approached as entity matching or record linkage in natural language processing (NLP) where the idea is to match a query address to a reference set of addresses with known geocodes (Comber and Arribas-Bel, 2019; Lin et al., 2020; Li et al., 2022).

These models target for a fine-grained (i.e. building, campus level) or exact matching of addresses and thus do not serve to a large fraction of addresses in emerging geographies which have missing or inaccurate building/campus information. Figure 1 illustrates a conceptual view of our geocoding pipeline. Rather than matching a query address to an individual address in reference set, we aim to retrieve its nearest neighbours. Using neighbourhood based approach we attempt to treat addresses not in isolation but in the view of their multiple neighbours, making it more robust to inconsistencies in hard-to-resolve addresses. The matched set of addresses can be used to vend a geocode and/or to jointly form a neighbourhood for the query address which can be vended as a approximate area to guide a delivery agent. In this paper, we focus on the experimental evaluation of geocoding task, and keep the neighbourhood polygons study as a work in the future.

We propose a novel deep metric learning based model to encode the geospatial distance semantics in address embeddings, in turn facilitating the retrieval of neighbours solely based on the address text. We pre-train the transformer based RoBERTa (Liu et al., 2019) model on address data and further employ a triplet network to learn quality address embeddings. A major proportion of hard-to-resolve addresses that this work targets, do not have any delivery history. Thus, our models draw supervision from past delivery scans while training only, and solely use address text (cold start) at the time of inference. In summary, our contributions are:

- We propose a deep metric learning based model to facilitate the encoding of geospatial distance semantics into address embeddings.
- We introduce a novel training data generation strategy to learn from geospatially rich addresses via weak supervision and transfer the knowledge to operate on cold-start addresses.
- To demonstrate the real-world impact of our model, we perform experiments on multiple emerging geographies (IN & the UAE).

This paper focuses on the downstream geocoding task, but learnt address embeddings can cater to other applications in the delivery planning space such as address correction, parsing, and learning neighbourhoods for package sorting.

2 Related Work

Short Text Geolocation Learning Geocoding short text (especially Tweets) has been an active area of research (Zheng et al., 2018). In (Hulden et al., 2015), authors propose a Naive Bayes classifier with kernel to learn the geospatial distribution of words and predict geolocation for tweets. (Paule et al., 2019) propose a weighted voting based nearest neighbours model to predict the location of traffic events. (Kulkarni et al., 2020) propose a neural network model with multi-level S2 (geospatial data structure) grids loss to learn tweets geolocation. Further (Qian et al., 2020) experiment with a seq2seq geocoding model to directly predict geohash string for Chinese addresses. (Li et al., 2019a) introduce GeoAttn model, which focuses on geolocation signals in the text and attends to the relevant Point-of-Interests (POIs) for location prediction. Although, most of these studies operate on a coarser level of geolocation (such as large geospatial grids, city) in contrast to the address geocoding task in e-commerce domain, which requires predicting within few meters of the customer doorstep to optimize delivery operations.

Entity Matching and Addresses In NLP, entity matching (or record linkage/deduplication) refers to the task of matching a query data instance to instances in a reference set (Hu et al., 2019). (Guo et al., 2016) propose a deep relevance matching model and more recently, the large language models for entity matching are explored by Ditto (Li et al., 2020) and dual objective fine-tuning of BERT (Peeters and Bizer, 2021).

Address geocoding has also been largely approached as entity matching task. (Comber and Arribas-Bel, 2019) propose to first parse the address text into address fields (unit, building, etc.), and then apply a pairwise matcher model to find a matching address in reference set and make a geocode prediction. (Lee et al., 2020) also implement a similar process where a rule-based parser and an SVM based matcher with building number interpolation are used for geocoding. Further, (Lin et al., 2020) and (Li et al., 2022) utilize deep learning based model for semantic matching of addresses. (Chen et al., 2021) propose a contrastive learning based address matcher for Chinese addresses while synthetically manipulating address texts to generate matching pairs. (Yang et al., 2019) propose to learn embedding for places and then uti-

lize them to train a supervised places deduplication model. In (Ganesan et al., 2021), authors propose a clustering based unsupervised model to learn POIs from the address data.

Deep Metric Learning Deep metric learning is being widely used on similarity retrieval tasks in both computer vision (CV) and NLP domains (Kaya and Bilge, 2019). (Hermans et al., 2017) apply triplet loss for person re-identification task and (Chen et al., 2020) introduce the contrastive learning of visual representations (SimCLR) for object detection. Sentence embeddings using siamese BERT networks are proposed to learn better downstream task specific embeddings (Reimers and Gurevych, 2019). SimCSE (Gao et al., 2021) and DeCLUTR (Gao et al., 2021) exploit contrastive learning to learn sentence representation in an unsupervised setting. In geospatial domain, Tile2Vec (Jean et al., 2019) and Hex2Vec (Woźniak and Szymański, 2021) explore embeddings learning of map tiles, whereas (Samano et al., 2020) explore the mobility data to learn regions representation.

To the best of our knowledge, none of the aforementioned studies target geocoding of hard-to-resolve addresses in emerging geographies. Further, a systematic way to impart geospatial distance semantics in address embeddings remains unexplored. Unstructured geographies pose a variety of challenges as discussed in the Section 1, making our contribution non-trivial and impactful.

3 Proposed Model

We adapt the K-Nearest Neighbours (K-NN) model (Altman, 1992) for the address geocoding task by using Kernel Density Estimation (KDE) (Parzen, 1962; Forman, 2021). Our workflow for geocode learning is illustrated in Figure 1. In essence, the K-NN model first retrieves the neighbourhood set \mathbb{N} for an address a and then predicts its geocode by picking the geocode of the neighbour x with highest kernel density value. Equation 1 formulates the kernel density estimator P over the retrieved neighbours \mathbb{N} where $K(x; h)$ is a Gaussian kernel with haversine metric. The bandwidth h works as a smoothing parameter, we chose h as 200 meters after manual validation over 25m to 400m.

$$P_h(x) = \frac{1}{|\mathbb{N}|h} \sum_{n \in \mathbb{N}} K(x - n; h) \quad (1)$$

The absolute nearest neighbours search becomes very computationally expensive in higher dimen-

sional input space. Thus, we employ approximate nearest neighbours search (Li et al., 2019b) and build an Annoy (Erik et al., 2018) index over the address embedding vectors to fetch neighbouring addresses from the reference set. One key difference here from the other address or entity matching systems (Lin et al., 2020; Li et al., 2022, 2020) is the flexibility as we are not restricting the match to a given building or campus, rather allowing a shallow matching on the full address text to arrive at a neighbourhood that can be of any size, shape and granularity. To this end, once the nearest neighbours are retrieved, we normalize their scores w.r.t. the maximum score and prune out the neighbours with low normalized score (below 0.25). This has an adaptive thresholding effect as all neighbours will be preserved if having more or less equal scores, and if there are disparity in scores then the low scored neighbours will get pruned. Also, we perform basic outlier removal of potentially incorrect neighbours via $mean \pm 2 * sd$ over latitude and longitude values to compute a neighbourhood polygon via convex hull. The geocode of the query address is computed using the described KDE model as a representative geocode of the neighbourhood. In this setting, quality representation of addresses are of utmost importance for retrieval of quality nearest neighbours. Thus, we propose a deep metric learning driven address representation learning approach in the following.

3.1 Deep Metric Learning

Deep distance metric learning (or simply, deep metric learning) aims to automatically construct task-specific distance metric from (weakly) supervised data by employing deep neural networks (Kaya and Bilge, 2019). The learned distance metric/pseudo-metric can then be used to perform various downstream tasks (e.g., information retrieval, clustering). In the context of addresses geocoding, the idealistic goal for the aforementioned neighbourhood retrieval problem is to fetch the true neighbours (i.e. to mimic geospatial distance semantics) for an address by using its text information only. Thus, we aim at learning an embedding transformation function $f_\theta(x) : R^I \rightarrow R^O$ which maps geospatially closer addresses from the input data manifold in R^I onto metrically close points in the output embedding space R^O (θ denotes parameter set). Similarly, f_θ should map geospatially far addresses in R^I onto metrically distant points in R^O .

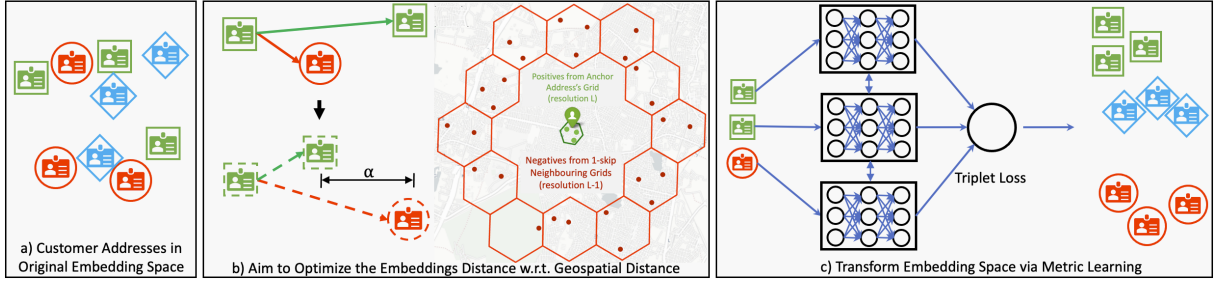


Figure 2: Deep metric learning on addresses to capture geospatial distance semantics

In the address domain, a key challenge with both the context-insensitive embeddings (e.g. FastText (Bojanowski et al., 2017)) or the contextualized embeddings (e.g. RoBERTa (Liu et al., 2019)) is the lack of understanding for geospatial distance semantics (cf. Section 4.2) as addresses do not follow a document or a paragraph like organization. Figure 2 depicts our adaptation of the deep metric learning workflow to learn quality address embeddings. As illustrated, we propose to systematically exploit geocodes of known addresses while training to give rise to geospatial distance semantics via weak supervision. To learn the transformation function f_θ , we choose RoBERTa as our base model as it has shown strong performance widely across multiple downstream NLP tasks (Liu et al., 2019).

Encoding Geospatial Semantics We employ contrastive learning approaches, specifically training via triplet loss. The triplet loss operates on triplets (x, x^+, x^-) of an anchor, a positive, and a negative instances. Equation 2 formulates the loss function with margin α and distance metric d . The objective here is to move the negative instance by distance margin α away from the anchor instance w.r.t. the positive instance. In our experiments, we chose margin 5 and Euclidean distance for triplet loss based on manual finetuning and practices in literature (Reimers and Gurevych, 2019).

$$L(x, x^+, x^-) = \max(0, d(f_\theta(x), f_\theta(x^+)) - d(f_\theta(x), f_\theta(x^-)) + \alpha) \quad (2)$$

3.2 Training Data Generation

In classification, supervised metric learning algorithms use instance class labels (e.g. object, face identity) to generate the training data. However, manually labeling the matching/non-matching address pairs is very expensive and unscalable task. We employ historical delivery scans data to automatically generate the weakly labeled training pairs

or the triplets. The address metric learning problem is now formulated as an optimization problem where we seek to find the parameters θ of function f_θ that optimize a objective function (i.e. triplet loss) measuring the agreement with training data.

Ideally, positive addresses for an address should be sampled from the absolute geospatial neighbours within some small β^+ distance and negatives should be sampled from the addresses which are relatively far β^- away. Here, the limitation is costly computation of haversine distance of each address to every other address, further even using some spatial data structure such as Ball Tree (Omohundro, 1989) involves significant computation overhead. To overcome this, we propose to use H3 geospatial indexing² system as an approximate solution to retrieve positive and negative addresses in a more intelligent manner. H3 is a hierarchical spatial data structure which subdivides the space into buckets of hexagonal grid shape. Every hexagonal grid has seven child grids below it in the hierarchy, thus, a hexagon of resolution L have 7 child hexagons of resolution $L + 1$ and so on (cf. Appendix C). For instance, $L = 10$ hexagon has edges of length 66m, and the children ($L = 11$) have 25m edges.

For an address, T positive addresses are sampled from its H3 grid of level L . T negative addresses are sampled from the ring of parent’s (i.e. level $L - 1$) 1-skip neighbouring grids as shown in Figure 2b. To this end, we generate triplets by varying the resolution ($L \in [11, 10, 9]$) for positive samples (and consequently for negatives). The motivation behind including triplets with varying resolution is to compile a more diverse training data where triplets can encode very a fine-grained as well as a coarse grained comparison of addresses. As addresses in close vicinity tend to differ only in the header part, we generate another T triplets where negatives are sampled from the city level to enforce sufficient focus on the tail address components.

²H3 geospatial index <https://github.com/uber/h3>

4 Experimental Evaluation

We evaluate the learnt embeddings intrinsically and on the downstream geocoding task.

4.1 Experimental Setup

We experiment with IN and the UAE addresses. For each of the dataset we use large unlabelled address text to pre-train the RoBERTa model and use historical delivery scans data to generate weak supervision for metric learning. We operate on the last few years of data which are worth hundreds of millions of shipments and tens of millions of unique addresses. We do minimal preprocessing of the address text by replacing repeated space and punctuations to single character. For evaluation, a few weeks of out-of-time network wide shipments are considered where learnt geolocations are compared against the observed delivery scans (marked by delivery associates). As our solution is targeted towards hard-to-resolve cases (i.e., production baselines couldn't vend any confident geocode and fall back to postal code/locality centroids), we only run our pipeline for this particular subset. Note that due to confidentiality reasons, we cannot reveal the actual proportion of hard-to-resolve addresses however, they are considerably high for the emerging geographies such as IN & the UAE, which is why improvements on this subset result in large amount of savings network-wide.

Deep Metric Learning Data Set For metric learning experiments, we only consider the addresses with at least H historical scans¹ to be more confident on the actual location. We take a stratified sample w.r.t. grids to have better representation of addresses across a geography and to not skew the learning disproportionately towards high density metropolitan areas. We generate $2 * T$ triplets¹ for an anchor address as explained in Section 3.2. To this end, we get a total of 37M triplets for IN and 7M triplets for the UAE.

Model Configurations and Baselines We perform extensive experiments on the task of geolocating hard-to-resolve addresses across various underlying models. We set up the current production geocoding system on the considered hard-to-resolve test set and report relative improvements over it. Due to the complex nature of these addresses, the baseline reduces to simply the centroid at postal code or locality level. For a better comparative analysis, we also consider a context-

insensitive model based on FastText, which is a skip-gram model trained with character n-grams of size 3-5 and window size of 8 for 10 epochs on address data. Among the transformer models, we have two groups – 1) The first group includes RoBERTa-General which is the general purpose English RoBERTa-base model, and RoBERTa-Address (6 layers) is trained from scratch on address data; 2) The second group is based on metric learning framework. RoBERTa-Triplet is trained on triplets generated by sampling positives at single fixed H3 resolution ($L = 11$) only and negatives are sampled only from the city level. In contrast, RoBERTa-Triplet-H3 is trained using the proposed training data generation strategy, which operates at multiple H3 resolutions to generate better quality triplets (cf. Section 3.2). These two models are fine-tuned over RoBERTa-Address. The final address embedding vector is computed via mean pooling over token embeddings of the final layer.

Pre-training Address Language Model As addresses have quite different vocabulary and domain than general English text, we train from scratch the geography specific RoBERTa models (6 layers) with masked language modeling (MLM) objective on addresses data (tens of millions). We train Byte-Pair Encoding tokenizers with vocabulary size of 52K. The model training with sequence length of 100 and batch size of 64 for 10 epochs takes around 49 hours on 4 Tesla V100 GPUs.

4.2 Assessing Embeddings Quality

To intrinsically measure the geospatial distance semantics captured in address embeddings, we compute cosine similarity co-relation on address pairs. A test set of 0.5M pairs is compiled by sampling positive pairs (score 1) from the same H3 grid of resolution 9 and negatives (score 0) are sampled from city level (equal + & - pairs). Further, to evaluate more complex relationship among addresses, we generate a set of 0.5M triplets constrained by

Model	Pearson		Triplet Acc	
	IN	UAE	IN	UAE
FastText	0.56	0.66	84.23	86.91
RoBERTa-General	0.35	0.45	70.42	75.76
RoBERTa-Address	0.63	0.68	85.78	87.02
RoBERTa-Triplet	0.76	0.75	91.33	91.79
RoBERTa-Triplet-H3	0.81	0.84	93.92	95.54

Table 1: Address pairs cosine similarity co-relation (cf. Appendix A for density plots) and Triplet accuracy

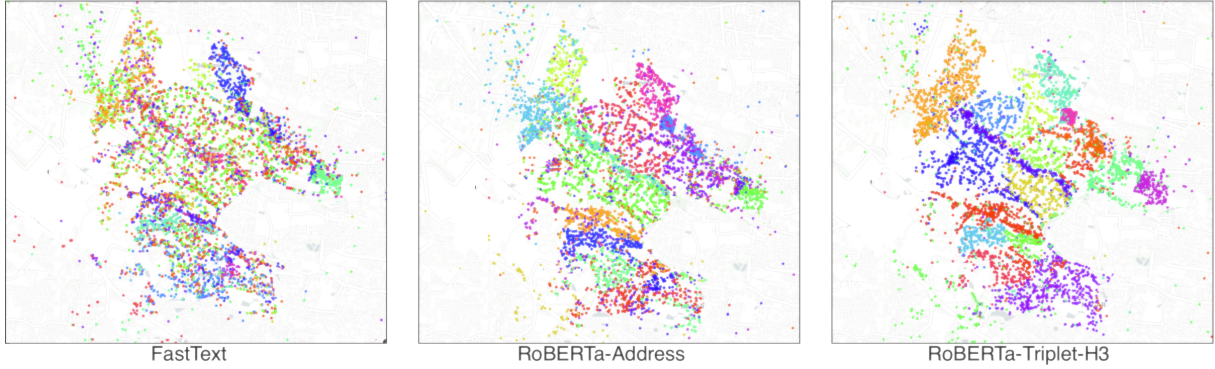


Figure 3: Clustering the addresses using different embeddings and visualizing via their geocodes (Note: background maps are modified and blurred to preserve customers’ privacy)

Geocoding Model	>Y DPMO (↓ %)		p25 (↓ %)		p50(↓ %)		p95(↓ %)	
	IN	UAE	IN	UAE	IN	UAE	IN	UAE
FastText	19.3%	48.9%	84.0%	94.3%	35.2%	82.4%	34.6%	61.6%
RoBERTa-General	9.8%	34.3%	52.1%	88.9%	9.1%	60.1%	-44.2%	31.8%
RoBERTa-Address	20.9%	47.4%	86.2%	94.0%	43.6%	80.2%	33.2%	60.5%
RoBERTa-Triplet	21.0%	52.1%	85.7%	95.5%	41.1%	86.7%	31.5%	60.1%
RoBERTa-Triplet-H3	22.0%	54.6%	88.6%	96.4%	42.8%	90.3%	32.2%	53.1%

Table 2: Geocoding metrics relative to the production baseline on shipments against hard-to-resolve addresses

the only condition that anchor will be geospatially closer to the positive than the negative. Then the triplet accuracy is computed to evaluate if embeddings pass the same criterion using cosine distance. Table 1 reports Pearson co-relation and the triplet accuracy metrics and we observe that the metric learning based models outperform others by a large margin (cf. Appendix A for density plots).

We also do a qualitative analysis by clustering (K-means with K=20) the addresses using their embeddings and visualizing them via their geocodes (cf. Figure 3 for 50K addresses in an IN postal code). The motivation is that embeddings which capture quality geospatial distance semantics will result in smoother clusters by facilitating the grouping of geospatially closer addresses. We observe that FastText based embeddings produce clusters with very high overlaps. In contrast, RoBERTa-Triplet-H3 embeddings facilitate smoother boundary clusters because of better geospatial distance semantic understanding. RoBERTa-Triplet-H3 embeddings clusters’ quality can also be seen slightly improving over the RoBERTa-Address. This is also visible in Silhouette scores which are 0.02, 0.08, and 0.13 for FastText, RoBERTa-Address, and RoBERTa-Triplet-H3 respectively. The observed geospatial distance semantics are beneficial for multiple downstream tasks such as address correction, package sortation, and address geocoding.

4.3 Geolocating Hard-to-Resolve Addresses

We compute neighbourhood level geocodes via KDE over the retrieved neighbours as illustrated in Figure 1 and serve to guide the drivers to a closer proximity in the absence of any better geocode. Table 2 presents experimental results via various geocoding metrics relative to the production baseline on shipments for the chosen test period. DPMO (Defects Per Million Opportunities) measures the number of prediction falling beyond Y^1 meters normalized to a million. The percentile metrics (p25, p50, p95) capture the distribution of error distances (actual vs predicted geocode) on the test set. A superior model shall lead to higher reductions in these metrics.

It can be observed from Table 2 that the proposed model based on deep metric learning outperforms the production baseline by a substantial margin as well as stands superior in comparison to other baselines i.e. FastText and basic Transformer models. The poor performance of RoBERTa-General model is due to its training on general purpose English text only. It can also be seen that RoBERTa-Triplet-H3 improves over RoBERTa-Triplet by a large margin, which can be directly attributed to the importance of our proposed training data generation strategy. Overall, we observe an improvement of 22% in DPMO for IN and 54% for the UAE (cf. Appendix B for geocoding anecdotes). This reduction in num-

ber of defects is directly translatable to the saved operational cost arising from delivery defects. It should be noted that addresses in IN & the UAE are quite different in nature, thus, improved metrics confirm the wide applicability of our framework. Further, IN has much bigger scale and more diverse addresses than the UAE, which manifests in our results with larger improvements in the UAE. We performed an ablation study by experimenting without adaptive thresholding (cf. Section 3) and observed degraded performance across models (IN DPMO became 13% for FastText and 16% for RoBERTa-Triplet-H3). It is also worth pointing out that the proposed model is trained with weak supervision and does not have a dependency on any manually curated ground truth or the address parsing models.

5 Conclusion and Future Work

In this work, we presented an efficient nearest neighbours & deep metric learning based approach to perform the address geocoding and facilitate the capturing of geospatial distance semantics in address embeddings. We intrinsically observe quantifiable improvements in address embeddings quality. Encouraging results from offline experiments suggest an immediate improvement in serving hard-to-resolve addresses. Our model operates solely using address text at the inference time, and is trained without any manually curated labels making it scalable across emerging geographies such as IN, the UAE, Mexico, and Saudi Arabia.

We plan to perform online experiments and extend our models to multi-lingual addresses in order to deal with prevalent issues like code switching in emerging geographies. We also would like to enhance our negative mining strategies and explore a pairwise cross encoder model to filter out the poorly retrieved neighbours. Retrieval of addresses from the neighbourhood can power many other downstream applications such as address component correction, address auto-complete suggestions, and optimizing delivery station assignment. We plan to explore geospatial constraints aware neighbourhood learning (e.g., to ensure neighbourhoods do not cross natural obstacles such as water bodies and highways).

References

Naomi S Altman. 1992. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

- 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.
- Jian Chen, Jianpeng Chen, Xiangrong She, Jian Mao, and Gang Chen. 2021. Deep contrast learning approach for address semantic matching. *Applied Sciences*, 11(16):7608.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. [A simple framework for contrastive learning of visual representations](#). *CoRR*, abs/2002.05709.
- Sam Comber and Daniel Arribas-Bel. 2019. [Machine learning innovations in address matching: A practical comparison of word2vec and crfs](#). *Transactions in GIS*, 23(2):334–348.
- Bernhardsson Erik et al. 2018. Annoy (approximate nearest neighbors oh yeah): Approximate nearest neighbors in c++/python optimized for memory usage and loading/saving to disk. <https://github.com/spotify/annoy>.
- George Forman. 2021. Getting your package to the right place: Supervised machine learning for geolocation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 403–419. Springer.
- Abhinav Ganesan, Anubhav Gupta, and Jose Mathew. 2021. [Mining points of interest via address embeddings: An unsupervised approach](#). In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Location-Based Recommendations, Geosocial Networks and Geoadvertising, LocalRec '21*, New York, NY, USA. Association for Computing Machinery.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.
- Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 55–64.
- Alexander Hermans, Lucas Beyer, and Bastian Leibe. 2017. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*.
- Weiwei Hu, Anhong Dang, and Ying Tan. 2019. A survey of state-of-the-art short text matching algorithms. In *Data Mining and Big Data*, pages 211–219, Singapore. Springer Singapore.
- Mans Hulden, Miikka Silfverberg, and Jerid Francom. 2015. Kernel density estimation for text-based geolocation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

- Neal Jean, Sherrie Wang, Anshul Samar, George Azzari, David Lobell, and Stefano Ermon. 2019. Tile2vec: Unsupervised representation learning for spatially distributed data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3967–3974.
- Mahmut Kaya and Hasan Şakir Bilge. 2019. Deep metric learning: A survey. *Symmetry*, 11(9):1066.
- Sayali Kulkarni, Shailee Jain, Mohammad Javad Hosseini, Jason Baldridge, Eugene Ie, and Li Zhang. 2020. Spatial language representation with multi-level geocoding. *arXiv preprint arXiv:2008.09236*.
- Kangjae Lee, Alexis Richard C Claridades, and Jiyeong Lee. 2020. Improving a street-based geocoding algorithm using machine learning techniques. *Applied Sciences*, 10(16):5628.
- Fangfang Li, Yiheng Lu, Xingliang Mao, Junwen Duan, and Xiyao Liu. 2022. Multi-task deep learning model based on hierarchical relations of address elements for semantic address matching. *Neural Comput. Appl.*, 34(11):8919–8931.
- Sha Li, Chao Zhang, Dongming Lei, Ji Li, and Jiawei Han. 2019a. *GeoAttn: Localization of Social Media Messages via Attentional Memory Network*, pages 64–72. Proceedings of the 2019 SIAM International Conference on Data Mining (SDM).
- Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019b. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488.
- Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.
- Yue Lin, Mengjun Kang, Yuyang Wu, Qingyun Du, and Tao Liu. 2020. A deep learning architecture for semantic address matching. *International Journal of Geographical Information Science*, 34(3):559–576.
- 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*.
- Stephen M. Omohundro. 1989. Five balltree construction algorithms. Technical report.
- Emanuel Parzen. 1962. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076.
- Jorge David Gonzalez Paule, Yeran Sun, and Yashar Moshfeghi. 2019. On fine-grained geolocalisation of tweets and real-time traffic incident detection. *Information Processing & Management*, 56(3):1119–1132.
- Ralph Peeters and Christian Bizer. 2021. Dual-objective fine-tuning of bert for entity matching. *Proceedings of the VLDB Endowment*, 14(10):1913–1921.
- Chunyao Qian, Chao Yi, Chengqi Cheng, Guoliang Pu, and Jiashu Liu. 2020. A coarse-to-fine model for geolocating chinese addresses. *ISPRS International Journal of Geo-Information*, 9(12):698.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.
- Noe Samano, Mengjie Zhou, and Andrew Calway. 2020. You are here: Geolocation by embedding maps and images. In *Computer Vision – ECCV 2020*, pages 502–518, Cham. Springer International Publishing.
- Szymon Woźniak and Piotr Szymański. 2021. *Hex2vec: Context-Aware Embedding H3 Hexagons with OpenStreetMap Tags*, page 61–71. Association for Computing Machinery, New York, NY, USA.
- Carl Yang, Do Huy Hoang, Tomas Mikolov, and Jiawei Han. 2019. Place deduplication with embeddings. In *The World Wide Web Conference*, pages 3420–3426.
- Xin Zheng, Jialong Han, and Aixin Sun. 2018. A survey of location prediction on twitter. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1652–1671.

A Cosine Similarity Density Plots

Figure 4 depicts the cosine similarity density plots for positive (label 1) and negative (label 0) address pairs in the test set (cf. Section 4.2). The x-axis represents cosine similarity values and it can be seen in Figure 4 that RoBERTa-Triplet-H3 model segregates well positives from negatives with least overlap between the two density curves (cf. Fig. 4c) in comparison to others (cf. Fig. 4a,b).

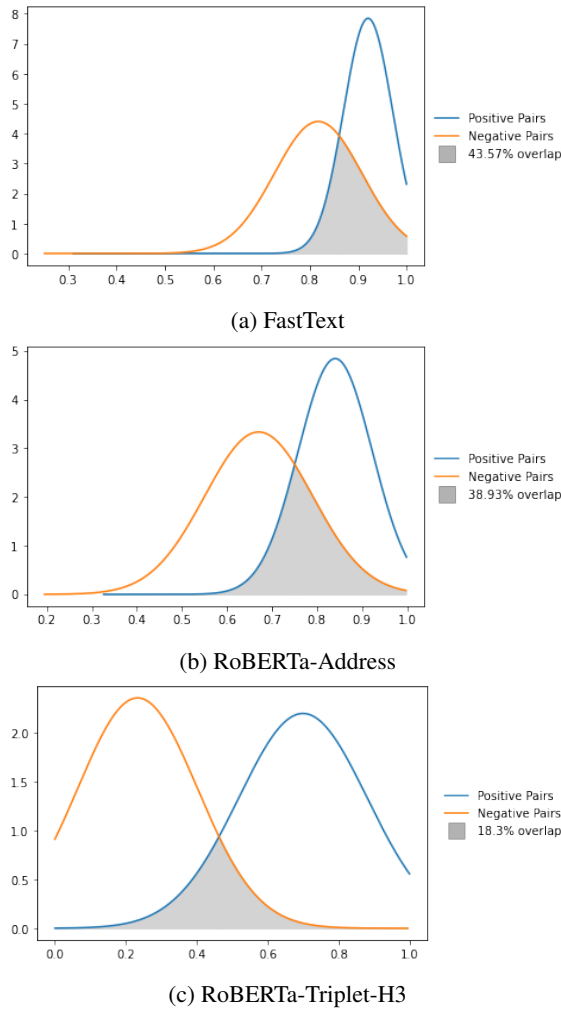


Figure 4: Density curves of positive (label=1) and negative (label=0) address pairs

B Anecdotes on Geocoding

Figure 5 depicts geocodes of the retrieved neighbours (along with neighbourhood polygons/circles) and the predicted geocode by various models for a hard-to-resolve address $X-X-X$, *Shivalayam Nagar*, 500070, *Hyderabad*, *Telangana*. The masked information ($X-X-X$) here is the house number, which carry some relevance for geocoding but usually noisy and do not follow a standard pattern. It

is a hard address as it is relatively sparse with no landmark information and the locality name is misspelled (*Shivalayam* instead of *Sachivalayam*). *Shivalayam* means Temple whereas *Sachivalayam* means Government Admin Office. There exist no *Shivalayam Nagar* in 500070, *Hyderabad*. We observe that the FastText model struggles to retrieve good quality neighbour addresses. RoBERTa-Address model utilizes the context and retrieves few good addresses but at the same time many poor matches too. The RoBERTa-Triplet-H3 model utilizes the contextual information best along with house number in address header to be resilient towards wrong locality name. It produces a quality set of neighbouring addresses to bring the predicted geocode as close as 39m to the actual location.

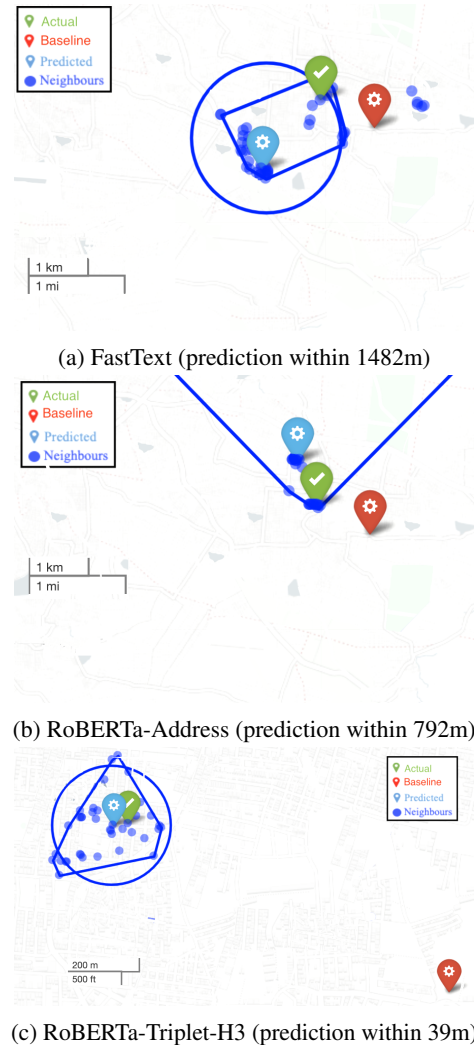


Figure 5: Retrieved nearest neighbour addresses by various models for an example address: $X-X-X$ *Shivalayam Nagar*, 500070, *Hyderabad*, *Telangana*. The current production baseline vends a geocode 986m away and we vend within 39m. (Note: background maps are modified and blurred to preserve customers' privacy)

C H3 Hexagonal Grids

Table 3 reports size of hexagon grids with respect to various H3 index resolution levels, and Figure 6 illustrates the hierarchical relation between grids. The geographical containment of children by a parent is approximate while the logical containment in the index is exact. We choose H3 over other geospatial indices such as Geohash because of the benefits observed via the symmetry of hexagonal shape in contrast to squares/triangles which have neighbors at varying distances in different directions.



Figure 6: H3 parent and child hexagonal grids hierarchy

H3 Resolution	Edge (meters)	Diagonal (meters)
0	11,07,712.6	22,15,425.2
1	4,18,676.0	8,37,352.0
2	1,58,244.7	3,16,489.3
3	59,810.9	1,19,621.7
4	22,606.4	45,212.8
5	8,544.4	17,088.8
6	3,229.5	6,459.0
7	1,220.6	2,441.3
8	461.4	922.7
9	174.4	348.8
10	65.9	131.8
11	24.9	49.8
12	9.4	18.8
13	3.6	7.1
14	1.3	2.7
15	0.5	1.0

Table 3: H3 hexagonal grid edge and diameter sizes w.r.t. the resolution levels

Meta-learning Pathologies from Radiology Reports using Variance Aware Prototypical Networks

Arijit Sehanobish Kawshik Kannan* Nabila Abraham Anasuya Das Benjamin Odry

Covera Health

New York City, NY

{arijit.sehanobish, kawshik.kannan, nabila.abraham,
anasuya.das, benjamin.odry}@coverahealth.com

Abstract

Large pretrained Transformer-based language models like BERT and GPT have changed the landscape of Natural Language Processing (NLP). However, fine tuning such models still requires a large number of training examples for each target task, thus annotating multiple datasets and training these models on various downstream tasks becomes time consuming and expensive. In this work, we propose a simple extension of the Prototypical Networks for few-shot text classification. Our main idea is to replace the class prototypes by Gaussians and introduce a regularization term that encourages the examples to be clustered near the appropriate class centroids. Experimental results show that our method outperforms various strong baselines on 13 public and 4 internal datasets. Furthermore, we use the class distributions as a tool for detecting potential out-of-distribution (OOD) data points during deployment.

1 Introduction

Pretrained Transformer-based language models (PLMs) have achieved great success on many NLP tasks (Devlin et al., 2019; Brown et al., 2020), but still need a large number of in-domain labeled examples for finetuning (Yogatama et al., 2019). Learning to learn (Lake et al., 2015a; Schmidhuber, 1987; Bengio et al., 1997) from limited supervision is an important problem with widespread application in areas where obtaining labeled data can be difficult or expensive. To that end, meta-learning methods have been proposed as effective solutions for few-shot learning (Hospedales et al., 2020). Current applications of such meta-learning methods have shown improved performance in few-shot learning for vision tasks such as learning to classify new image classes within a similar dataset. Namely, on classical few-shot image classification benchmarks, the training tasks are sampled from

a “single” larger dataset (for ex: Omniglot (Lake et al., 2015b) and miniImageNet (Vinyals et al., 2016)), and the label space contains the same task structure for all tasks. There has been a similar trend of such classical methods in NLP as well (Geng et al., 2019). In contrast, in text classification tasks, the set of source tasks available during training and target tasks during evaluation can range from sentiment analysis to grammatical acceptability judgment (Bansal et al., 2020a,b). In recent works (Wang et al., 2021), the authors use a range of different source tasks (different not only in terms of input domain, but also their task structure i.e. label semantics, and number of labels) for meta-training and show successful performance on a wide range of downstream tasks. In spite of this success, meta-training on various source tasks is quite challenging as it requires resistance to overfitting to certain source tasks due to its few-shot nature and more task-specific adaptation due to the distinct nature among tasks (Roelofs et al., 2019).

However, in medical NLP, collecting large number of diverse labeled datasets is difficult. In our institution, we collect high quality labeled radiology reports (which are always used as held out test data) and use it to train our internal annotators who then annotate our unlabeled data. This training process is expensive and time consuming. Our annotation process is described in section A. Thus a natural question is: if we have a large labeled dataset consisting of a lot of classes, can we use it to meta-train a model that can be used on a large number of downstream datasets where we have little to no training examples? This is a challenging problem as the reports can be structured differently based on the report type and there can be a substantial variation in writing style across radiologists from different institutions. Our main goal is to build out a set of extensible pipelines that can generalize to new pathologies typically in new sub-specialties while also generalizing across different health systems.

*Equal Contribution

In addition, the exact definition of the pathologies and their severity change can change depending on the clinical use case. This makes fully supervised approaches that rely on large labeled datasets expensive. Having few-shot capabilities allows us to annotate a handful of cases and rapidly expand the list of pathologies we can detect and classify. In addition, we can use our approach to generate pseudo labels for rare pathologies and enrich our validation and test sets for annotation by an in-house clinical team. Lastly our approach can be extended to support patient search and define custom cohorts of patients.

Our contributions in this work are the following: **(1)** We develop a novel loss function that extends the vanilla prototypical networks and introduce a regularization term that encourages tight clustering of examples near the class prototypes. **(2)** We meta-train our models on a large labeled dataset on shoulder MRI reports (single domain) and show good performance on 4 diverse downstream classification tasks on radiology reports on knee, cervical spine and chest. In addition to our internal datasets, we show superior performance of our method on 13 public benchmarks over well-known methods like Leopard. Our model is very simple to train, easy to deploy unlike gradient based methods and just requires a few additional lines of codes to a vanilla prototypical network trainer. **(3)** We deploy our system and use the dataset statistics to inform out-of-distribution (OOD) cases.

2 Related Work

There are three common approaches to meta-learning: metric-based, model-based, and optimization-based. Model agnostic meta-learning (MAML) (Finn et al., 2017) is an optimization-based approach to meta-learning which is agnostic to the model architecture and task specification. Over the years, several variants of the method have shown that it is an ideal candidate for learning to learn from diverse tasks (Nichol et al., 2018; Raghu et al., 2019; Bansal et al., 2020b). However, to solve a new task, MAML type methods would require training a new classification layer for the task. In contrast, metric-based approaches, such as prototypical networks (Vinyals et al., 2016; Snell et al., 2017), being non-parametric in nature can handle varied number of classes and thus can be easily deployed. Given the simple nature of prototypical networks, a lot of work has been

done to improve them (Allen et al., 2019; Zhang et al., 2019; Ding et al., 2022; Wang et al., 2021). Prototypical networks usually construct a class prototype (mean) using the support vectors to describe the class and, given a query example, assigns the class whose class prototype is closest to the query vector. In (Allen et al., 2019), the authors use a mixture of Gaussians to describe the class conditional distribution and in (Zhang et al., 2019); the authors try to model an unknown general class distribution. In (Ding et al., 2022), the authors use spherical Gaussians and a KL-divergence type function between the Gaussians to compute the function d in equation 2. However, the function used by the above authors is not a true metric, i.e. does not satisfy the triangle inequality. Triangle inequality is implicitly important since we use this metric as a form of distance which we optimize, so it makes sense to use a true metric. In this work we replace it by the Wasserstein distance which is a true metric and add in a regularization term that encourages the L_2 norm of the covariance matrices to be small, encouraging the class examples to be clustered close to the centroid. One of our main reasons to work with Gaussians is due to the closed form formula of the Wasserstein distance.

Few shot learning (FSL) in the medical domain has been mostly focused in computer vision (Singh et al., 2021). There are only a few works that have applied FSL in medical NLP (Ge et al., 2022) but most of those works have only focused on different tasks on MIMIC-III (Johnson et al., 2016) which is a single domain dataset (patients from ICU and one hospital system). To the best of our knowledge, ours is the first study to successfully apply FSL on a diverse set of medical datasets (diverse in terms of tasks and patient populations).

3 Datasets

All our internal datasets are MRI radiology reports detailing various pathologies in different body parts. Our models are meta-trained on a dataset of shoulder pathologies which is collected from 74 unique and de-identified institutions in the United States. 60 labels are chosen for training and 20 novel labels are chosen for validation. The number of training labels is similar to some well-known image datasets (Lake et al., 2015b; Vinyals et al., 2016; Wah et al., 2011). This diverse dataset has a rich label space detailing multiple structures in shoulder, granular pathologies and their severity

levels in each structure. The relationship between the granularity/severity of these pathologies at different structures can be leveraged for other pathologies in different body parts and may lead to successful transfer to various downstream tasks. The labels are split such that all pathologies in a given structure appear at either training or validation but not both. More details about the label space can be found in section B. The figure 1 and the table 1 shows the distribution of labels and an example of this dataset can be found in figure 4. Our met-

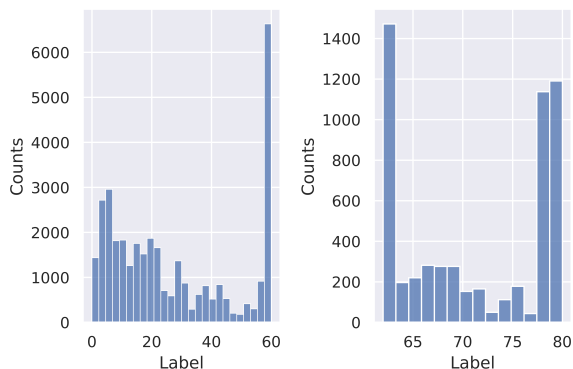


Figure 1: Histogram showing the label distribution in (left) train and (right) validation dataset.

learner is applied to 4 downstream binary classification tasks spanning different sub-specialities (cancer screening, musculoskeletal radiology, and neuro-radiology) that are both common as well as clinically important. The statistics for each task are given in table 2 : **(1)** High risk cancer screening for lung nodules (according to Fleischner guidelines (Nair et al., 2018) which bucket patients at high-risk of lung cancer and requiring follow up imaging immediately or within 3 months as belonging to Category High Risk ; we consider patients not at high-risk as Low Risk), **(2)** Complete Anterior Cruciate Ligament (ACL) tear (Grade 3) vs not Complete ACL tear, **(3)** Acute ACL tears (MRI examination was performed within 6 weeks of injury) and typified by the presence of diffuse or focal increased signal within the ligament vs not Acute ACL tear (Dimond et al., 1998), **(4)** Severe vs not severe neural foraminal stenosis in the cervical spine as severe foraminal stenosis may indicate nerve impingement, which is clinically significant. Acute tear in ACL refers to the age of the tear/injury whereas the complete tear refers to the integrity of the ligament. Our testing datasets are diverse and sampled from different institutions: the knee data, lung dataset and cervical dataset is sampled

Split	Number of examples	Min	Max	Average
Train	34595	79	6379	567
Validation	5754	44	1138	303

Table 1: Statistics of our meta-training and meta-validation dataset, where the min/max/average refer to min/max/average examples per label.

from 50, 4 and 65 institutions respectively and our annotation process is described in Appendix A. Examples of these datasets can be found in figure 10 (knee), figure 6 (lung), and figure 8 (cervical).

Task	Validation Distribution	Test Distribution
Lung Nodule	Low Risk : 233	Low Risk : 347
	High Risk : 30	High Risk : 46
Knee ACL	Normal: 258	Normal : 439
Acute Tear	Acute Tear: 48	Acute Tear: 93
Knee ACL	Normal : 263	Normal : 429
Complete Tear	Complete Tear : 44	Complete Tear :103
Neural Foraminal	Normal : 215	Normal : 789
Stenosis	Abnormal : 43	Abnormal : 91

Table 2: Statistics of our downstream testing datasets

4 Workflow

Our workflow consists of the following parts: A

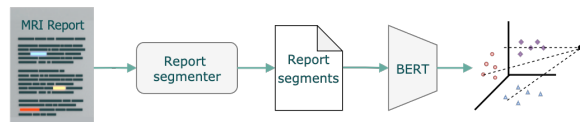


Figure 2: Overview of our workflow. A report is passed through a report segmenter which splits it into sentences and extracts the relevant portion of the text for downstream classification. The relevant text is passed through our model and we use the pre-computed prototypes and class variances to assign a label to the query point.

report is first de-identified according to HIPAA regulations and passed through a sentence parser (ex. Spacy (Honnibal et al., 2020)) that splits the report into sentences. In the shoulder dataset, each of these sentences is labeled with the appropriate structure and severity label and we filter out sentences that do not have such a label. We first train a meta-learner in an episodic fashion on this dataset and choose the best model based on meta-validation accuracy.

For our downstream tasks, we use a body-part specific custom data processor to collect sentences related to a given structure (ACL in knee, different vertebrae in the cervical spine, the entire impression section for lung reports) and concatenate them

together to create a paragraph describing all the pathologies in the structure of interest. Detailed description of preprocessing for different body parts, is presented in Appendix C. The concatenated text from the validation sets of each task is passed to our trained meta-learner to generate the relevant class statistics (mean and variance). We then perform pathology classification on the test set by using our trained meta-learner and the saved class statistics. The downstream tasks are similar to the shoulder task in the sense that the pathology classification is performed on a sequence of sentences that all pertain to the same anatomical structure. Thus our approach needs to learn the language that describes the severity of the pathology for a specific anatomical structure.

We would like to shed some light on the complexity of the language we encounter. Since our dataset is sourced from multiple health systems, and not all reports follow a standard structure, there is a large amount of variation in the language describing the same diagnosis. For example: a severe tear can be referred to as a rupture, or only the size of the nodule is mentioned without specifying that it is low risk (see Appendix C for more examples). Furthermore, most of our pipelines attempt to classify the different severities for a given pathology and the language describing severity can vary. While it might be possible to construct a rule based system to extract the diagnoses and severities we are interested in, it will be difficult to generalize as we expand to more diagnoses as well as to new health systems.

5 Prototypical Networks

Prototypical Networks or ProtoNets (Snell et al., 2017) use an embedding function f_θ to encode each input into a M -dimensional feature vector. A prototype is defined for every class $c \in \mathcal{L}$, as the mean of the set of embedded support data samples (S_c) for the given class, i.e.

$$v_c = \frac{1}{|S_c|} \sum_{(x_i, y_i) \in S_c} f_\theta(x_i). \quad (1)$$

The distribution over classes for a given test input x is a softmax over the inverse of distances between the test data embedding and prototype vectors.

$$\begin{aligned} P(y = c|x) &= \text{softmax}(-d(f_\theta(x), v_c)) \\ &= \frac{\exp(-d(f_\theta(x), v_c))}{\sum_{c' \in \mathcal{L}} \exp(-d(f_\theta(x), v_{c'}))} \end{aligned} \quad (2)$$

where d can be any (differentiable) distance function. The loss function is negative log-likelihood:

$$\mathbb{L}(\theta) = -\log P_\theta(y = c|x).$$

ProtoNets are simple and easy to train and deploy. The mean is used to capture the entire conditional distribution $P(y = c|x)$, thus losing a lot of information about the underlying distribution. A lot of work (Ding et al., 2022; Allen et al., 2019; Zhang et al., 2019) has focused on improving ProtoNets by taking into account the above observation. We extend ProtoNets by incorporating the variance (2nd moment) of the distribution and use distributional distance, i.e. 2-Wasserstein metric, directly generalizing the vanilla ProtoNets.

5.1 Variance Aware ProtoNets

In this work, we model each conditional distribution as a Gaussian. Now the main question is: *how do we match a query example with a distribution?* The simplest thing here is to treat the query example as a Dirac distribution. With that formulation in mind, recall: the Wasserstein-Bures metric between Gaussians (m_i, Σ_i) is given by:

$$d^2 = \|m_1 - m_2\|^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2(\Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}})^{\frac{1}{2}})$$

Given $(x_i, y_i) \in S_c$, where S_c is the support set of examples belonging to class c , we compute the mean m_c and covariance matrix Σ_c ; the computation of Wasserstein distance between a Gaussian and a query vector q (i.e. a Dirac) boils down to

$$d^2 = \|m_c - q\|^2 + \text{Tr}(\Sigma_c) \quad (3)$$

The above formula shows that we can simplify our conditional distribution to be a Gaussian with a *diagonal* covariance matrix. This brings down our space complexity to store this covariance matrix from $O(n^2)$ to $O(n)$. Note, this is a direct generalization of the vanilla prototypical networks as the vanilla prototypical networks can be interpreted as computing the Wasserstein distance (aka simple L_2 distance) between two Dirac distributions (mean of the conditional distribution and the query sample). We also propose another variant of the above called Isotropic Gaussian variant where we average over the diagonal entries of Σ_c , i.e. $\alpha = \frac{1}{n}(\Sigma_c)_{ii}$ and redefine $\Sigma_c = \alpha I$, where I is the identity matrix, allowing us to just store the scalar α , further reducing the space complexity. Furthermore, we

Backbone	Methods	Foraminal	Knee (Acute Tear vs Not)	Knee (Complete tear vs Not)	Lung
PubMedBERT	Baseline	0.38	0.44	0.49	0.36
	Multi-Task	0.41	0.47	0.52	0.39
	Vanilla ProtoNet	0.79	0.73	0.60	0.68
	Big ProtoNet	0.58	0.59	0.51	0.64
	Leopard	0.84	0.78	0.80	0.74
	ProtoNet w/ Isotropic Gaussian	0.81	0.74	0.76	0.69
	ProtoNet w/ Isotropic Gaussian + reg	0.83	0.76	0.77	0.73
	Variance Aware ProtoNet (ours)	0.84	0.78	0.79	0.76
	Variance Aware ProtoNet + reg (ours)	0.86	0.81	0.84	0.80
	PubMedBERT w/ Adapters	Baseline	0.42	0.47	0.51
Multi-Task		0.44	0.49	0.53	0.43
Vanilla ProtoNet		0.78	0.71	0.69	0.66
Big ProtoNet		0.59	0.57	0.54	0.67
ProtoNet w/ Isotropic Gaussian		0.83	0.75	0.78	0.72
ProtoNet w/ Isotropic Gaussian + reg		0.89	0.80	0.86	0.77
Variance Aware ProtoNet (ours)		0.87	0.77	0.81	0.74
Variance Aware ProtoNet + reg (ours)		0.91	0.82	0.89	0.78

Table 3: Table showing F1 scores of Few Shot Models in downstream classification tasks.

regularize the negative log likelihood loss to prevent the variance term from blowing up. Our new loss function reads:

$$\mathcal{L}(\theta) = \mathbb{L}(\theta) + \frac{\lambda}{\text{ways}} \|\Sigma_c\|_F \quad (4)$$

where ways are the number of classes in the mini-batch and $\|\cdot\|_F$ is the Frobenius norm and we average the norm of the variance matrix over all the classes in a given meta-batch. The extra regularization term is designed to encourage the examples to be close to the appropriate cluster centroid. This term can also be seen as an entropic regularization term, i.e. up to a factor as the exponential of $KL(p||q)$, where $p = N(m_c, \Sigma_c)$ and $q = N(m_c, I)$. This type of entropy regularized Wasserstein distances is widely studied (Cuturi and Doucet, 2014; Altschuler et al., 2021).

A PyTorch style pseudocode is described in Algorithm 1, where the teal color refers to the changes to a vanilla prototypical networks trainer. We provide detailed motivation for using Wasserstein distance instead of KL divergence in section E.2. This also explains why we compute the Wasserstein distance between the query and the estimated class distribution instead of a simple likelihood.

6 Experiments

All our experiments are run on 4 V100 16 GB GPU using PyTorch (Paszke et al., 2019) and HuggingFace libraries (Wolf et al., 2020). Bert-base (Devlin et al., 2019), Clinical BERT (Alsentzer et al., 2019) and PubMedBERT (Gu et al., 2021) are used as our backbone models. Adapters (Pfeiffer et al., 2020) are applied to each of these backbone models.

While training adapter based models, the BERT weights are frozen and only the adapter weights are updated, thus requiring less resources to train. This idea is similar to (Raghu et al., 2019) in the sense that we are reusing the features from these deep pre-trained models. We compare our methods to Leopard (Bansal et al., 2020a), vanilla ProtoNets and big ProtoNets (Ding et al., 2022). Additional results with BERT-base and Clinical BERT backbones can be found in table 6 and table 7. Meta-training is done in an episodic manner using 4-way 8-shot and 16-examples as support. For meta-training on the shoulder dataset, we set the variance regularizer hyperparameter to be .1. It is an important hyperparameter and detailed ablation study is conducted in section E.1. Other hyperparameters and design choices are described in section E.

To prevent overfitting on the test set, we choose the best model from each of these experiments based on the meta-validation accuracy and apply it to our downstream classification tasks. We note that these downstream tasks are significantly different from the few shot regime these models are trained in. Moreover for these downstream tasks, we train BERT models on each task and a multi-tasking model to provide additional baselines.

In all our experiments, PubMedBERT consistently outperforms BERT-base and Clinical BERT by an average of 5 points and 3 points respectively. We believe the reason behind the improved performance is the domain specific vocabulary. Even though Clinical BERT is pre-trained on MIMIC-III (Johnson et al., 2016), it still shares the same vocabulary as BERT-base.

ProtoNet-BERT shows better performance and

faster convergence rates during training and validation (Table 4), but it is outperformed by ProtoNet-AdapterBERT which has fewer orders of magnitude of parameters to learn (Table 3). Like (Wang et al., 2021) we believe that ProtoNet-BERT is more vulnerable to overfitting on the meta-training tasks than the ProtoNet-AdapterBERT. Finally, we note that even though Big ProtoNets work well on meta-validation, they fail on our downstream tasks. We hypothesize that it is due to the fact that big protonets are encouraged to have large radii which has the potential to become a bottleneck where the data distribution is highly imbalanced causing the spherical Gaussians to overlap. In fact, we have found that doing the exact opposite (i.e. constricting the norms of the covariance matrix), tends to produce better results. Finally instead of using

Backbone	Methods	Accuracy
PubMedBERT	Vanilla ProtoNet	89.1 ± 1.1
	Big ProtoNet	90.8 ± 1.2
	Leopard	85.1 ± 9.2
	ProtoNet w/ Isotropic Gaussian	90.2 ± 1.4
	ProtoNet w/ Isotropic Gaussian + reg	92.1 ± .8
	Variance Aware ProtoNet (ours)	91.5 ± 1.3
	Variance Aware ProtoNet + reg (ours)	92.9 ± .9
PubMedBERT w/Adapters	Vanilla ProtoNet	88.3 ± 1.4
	Big ProtoNets	89.4 ± 1.2
	ProtoNet w/ Isotropic Gaussian	89.8 ± 1.4
	ProtoNet w/ Isotropic Gaussian + reg	90.9 ± .7
	Variance Aware ProtoNet (ours)	90.5 ± 1.3
	Variance Aware ProtoNet + reg (ours)	91.2 ± .8

Table 4: Results showing accuracy percentages on the meta-validation dataset. We sampled 1000 tasks with 4-way 8-shot and 16-support classification. We replicate each experiment over 10 random seeds.

the entire validation set to compute the class distribution, we also experiment with choosing a k shots from the validation set to compute the class distribution (figure 12 in section G).

Our regularized Variance Aware ProtoNets with BERT-base + Adapter is also validated on 13 public datasets. For the models and datasets marked with * in table 5, we use the results reported in (Bansal et al., 2020a) and for those datasets, we use the code from (Wang et al., 2021) to generate the results for ProtoNet with Bottleneck Adapters while the rest of the results are taken from (Wang et al., 2021). The variance regularization hyperparameter is set to .01 for these experiments. Our method beats Leopard by **5**, **3** and **2** points on **4**, **8** and **16** shots, respectively. The training details for these experiments can be found in section F.

7 Deployment

Based on the results described in table 3, we choose to deploy our regularized Variance Aware ProtoNet with Adapter-PubMedBERT. Our pipeline is deployed on AWS using a single p3.2x instance housed with one NVIDIA V100 GPU. The main pipeline components include **(1)** body-part specific report segmenter, **(2)** PubMedBERT backbone with adapters and **(3)** a dictionary of class prototypes and class variances, for all classes in the datasets. On inference, requests sent to the pipeline include a body part which the pipeline utilizes to load up the relevant report segmenter, class prototypes and variances. A report is then ingested by the pipeline, parsed by a sentencizer, grouped into segments according to its body part specific segmentation, and then passed to the model. Class probabilities and labels are inferred after computing the Wasserstein distance between the text embedding and the appropriate class distributions. These outputs and pipeline metadata are written out to an AWS Redshift database cluster. The entire pipeline is orchestrated in batch mode with a large enough batch size to maximize GPU capacity resulting in an average latency of 68ms/report.

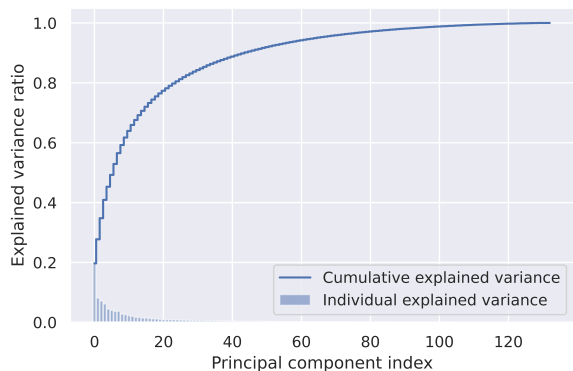


Figure 3: Variance along different directions for the Lung validation set

7.1 Monitoring

It is well-known that the BERT embeddings are highly anisotropic (Ethayarajh, 2019). We observe the same phenomenon in our meta-learned models as well (figure 3) which we use to our advantage to monitor OOD cases. For each class in a dataset, we pick top k -dimensions (a hyperparameter) of maximum variance. We then take the union of these indices that we call the set of dataset indices i.e. the indices that explain the variance among all classes

Shots	Dataset	BERT*	MT-BERT*	Leopard	ProtoNet	ProtoNet+Adapter	Variance Aware ProtoNet (Ours)
4	airline	42.76 ± 13.50	46.29 ± 12.26	54.95 ± 11.81	65.39 ± 12.73	65.33 ± 7.95	62.67 ± 11.18
	disaster	55.73 ± 10.29	50.61 ± 8.33	51.45 ± 4.25	54.01 ± 2.9	53.48 ± 4.76	53.89 ± 3.79
	emotion	9.20 ± 3.22	9.84 ± 2.14	11.71 ± 2.16	11.69 ± 1.87	12.52 ± 1.32	15.15 ± 4.19
	political_audience	51.89 ± 1.72	51.53 ± 1.80	52.60 ± 3.51	52.77 ± 5.86	51.88 ± 6.37	52.5 ± 6.45
	sentiment_kitchen*	56.93 ± 7.10	60.53 ± 9.25	78.35 ± 18.36	62.71 ± 9.53	83.13 ± 0.96	84.16 ± 1.37
	political_bias	54.57 ± 5.02	54.66 ± 3.74	60.49 ± 6.66	58.26 ± 10.42	61.72 ± 5.65	59.39 ± 6.18
	rating_electronics*	39.27 ± 10.15	41.20 ± 10.69	51.71 ± 7.20	37.40 ± 3.72	53.81 ± 6.01	55.49 ± 5.42
	political_message	15.64 ± 2.73	14.49 ± 1.75	15.69 ± 1.57	17.82 ± 1.33	20.98 ± 1.69	19.28 ± .91
	sentiment_books*	54.81 ± 3.75	64.93 ± 8.65	82.54 ± 1.33	73.15 ± 5.85	83.88 ± 0.55	84.95 ± 1.72
	rating_books*	39.42 ± 07.22	38.97 ± 13.27	48.44 ± 7.43	54.92 ± 6.18	59.20 ± 7.26	66.18 ± 7.89
	rating_dvd*	32.22 ± 08.72	41.23 ± 10.98	49.76 ± 9.80	47.73 ± 6.20	50.20 ± 10.26	52.59 ± 14.09
	rating_kitchen	34.76 ± 11.2	36.77 ± 10.62	50.21 ± 9.63	58.47 ± 11.12	55.99 ± 9.85	59.39 ± 8.79
	scitail*	58.53 ± 09.74	63.97 ± 14.36	69.50 ± 9.56	76.27 ± 4.26	77.84 ± 2.61	79.16 ± 2.54
	Average	41.98	44.23	52.11	51.58	56.15	57.29
8	airline	38.00 ± 17.06	49.81 ± 10.86	61.44 ± 3.90	69.14 ± 4.84	69.37 ± 2.46	69.31 ± 2.43
	disaster	56.31 ± 9.57	54.93 ± 7.88	55.96 ± 3.58	54.48 ± 3.17	53.85 ± 3.03	55.19 ± 2.77
	emotion	8.21 ± 2.12	11.21 ± 2.11	12.90 ± 1.63	13.10 ± 2.64	13.87 ± 1.82	15.1 ± 3.58
	political_audience	52.80 ± 2.72	54.34 ± 2.88	54.31 ± 3.95	55.17 ± 4.28	53.08 ± 6.08	53.82 ± 4.13
	sentiment_kitchen*	57.13 ± 6.60	69.66 ± 8.05	84.88 ± 1.12	70.19 ± 6.42	83.48 ± 0.44	84.69 ± .8
	political_bias	56.15 ± 3.75	54.79 ± 4.19	61.74 ± 6.73	63.22 ± 1.96	65.36 ± 2.03	64.09 ± .58
	rating_electronics*	28.74 ± 08.22	45.41 ± 09.49	54.78 ± 6.48	43.64 ± 7.31	56.97 ± 3.19	60.24 ± 2.62
	political_message	13.38 ± 1.74	15.24 ± 2.81	18.02 ± 2.32	20.40 ± 1.12	21.64 ± 1.72	20.44 ± 1.17
	sentiment_books*	53.54 ± 5.17	67.38 ± 9.78	83.03 ± 1.28	75.46 ± 6.87	83.9 ± 0.39	84.68 ± .85
	rating_books*	39.55 ± 10.01	46.77 ± 14.12	59.16 ± 4.13	52.13 ± 4.79	61.74 ± 6.83	65.54 ± 6.78
	rating_dvd*	36.35 ± 12.50	45.24 ± 9.76	53.28 ± 4.66	47.11 ± 4.00	53.25 ± 7.47	53.83 ± 10.46
	rating_kitchen	34.49 ± 8.72	47.98 ± 9.73	53.72 ± 10.31	57.08 ± 11.54	56.27 ± 10.70	56.68 ± 11.21
	scitail*	57.93 ± 10.70	68.24 ± 10.33	75.00 ± 2.42	78.27 ± 0.98	80.41 ± 1.05	80.57 ± .48
	Average	40.97	48.54	56.02	53.8	57.94	58.78
16	airline	58.01 ± 8.23	57.25 ± 9.90	62.15 ± 5.56	71.06 ± 1.60	69.83 ± 1.80	69.9 ± 1.06
	disaster	64.52 ± 8.93	60.70 ± 6.05	61.32 ± 2.83	55.30 ± 2.68	57.38 ± 5.25	60.14 ± 5.36
	emotion	13.43 ± 2.51	12.75 ± 2.04	13.38 ± 2.20	12.81 ± 1.21	14.11 ± 1.12	13.55 ± 3.51
	political_audience	58.45 ± 4.98	55.14 ± 4.57	57.71 ± 3.52	56.16 ± 2.81	57.23 ± 2.77	56.36 ± 2.29
	sentiment_kitchen*	68.88 ± 3.39	77.37 ± 6.74	85.27 ± 01.31	71.83 ± 5.94	83.72 ± 0.30	84.93 ± .49
	political_bias	60.96 ± 4.25	60.30 ± 3.26	65.08 ± 2.14	61.98 ± 6.89	65.38 ± 1.71	63.97 ± 2.49
	rating_electronics*	45.48 ± 06.13	47.29 ± 10.55	58.69 ± 2.41	44.83 ± 5.96	56.62 ± 5.62	61.01 ± 1.54
	political_message	20.67 ± 3.89	19.20 ± 2.20	18.07 ± 2.41	21.36 ± 0.86	24.00 ± 1.39	22.49 ± 1.31
	sentiment_books*	65.56 ± 4.12	69.65 ± 8.94	83.33 ± 0.79	77.26 ± 3.27	83.92 ± 0.41	84.91 ± 0.66
	rating_books*	43.08 ± 11.78	51.68 ± 11.27	61.02 ± 4.19	57.28 ± 4.57	64.75 ± 4.27	67.34 ± 7.52
	rating_dvd*	42.79 ± 10.18	45.19 ± 11.56	53.52 ± 4.77	48.39 ± 3.74	55.08 ± 4.92	56.63 ± 6.11
	rating_kitchen	47.94 ± 8.28	53.79 ± 9.47	57.00 ± 8.69	61.00 ± 9.17	59.45 ± 8.33	58.34 ± 11.72
	scitail*	65.66 ± 06.82	75.35 ± 04.80	77.03 ± 1.82	78.59 ± 0.48	80.27 ± .75	80.89 ± .23
	Average	50.42	52.74	57.97	55.22	59.36	60.04

Table 5: Results on some benchmark text datasets on a wide range of tasks from NLI, sentiment analysis and text classification. For the Variance Aware ProtoNet, we use BERT-base with bottleneck Adapters. For meta-training, WNLI (m/mm), SST-2, QQP, RTE, MRPC, QNLI, and SNLI datasets are used.

in the dataset. For any given query example, we compute the absolute difference (\vec{d}_j) between its embedding vector (\vec{q}) and class centroids (\vec{v}_j), i.e. the i -th coordinate $\vec{d}_j : \vec{d}_{j_i} = |\vec{q}_i - \vec{v}_{j_i}|$. We then select top k dimensions of the each of these \vec{d}_j . We propose an OOD metric called Average Variance Indices (AVI_k) by the overlap between the top-k difference vector indices and the top-k dataset indices, i.e. $AVI_k := \frac{|\cup_{j=1}^c \text{top-k}(\vec{d}_j)|}{\text{dataset indices}}$, where c is the number of classes. For ex: in case of the lung dataset: The text "*The heart is normal in size. There is no pericardial effusion. The pulmonary artery is enlarged.*" shows an AVI_10 score .79, whereas "*L1L2: There is no disc herniation in lumbar spine.*" gives a score of .31. As part of our monitoring, we threshold reports with an AVI_10 < .5 to further investigate if the report is OOD.

8 Conclusion

We extend Prototypical Networks by using Wasserstein distances instead of Euclidean distances and introduce a regularization term to encourage the class examples to be clustered close to the class prototype. By training our models on a label rich dataset (shoulder MRI reports), we show successful performance on a variety of tasks. Since the model weights are reused for all tasks, a single model is deployed enabling us to cut inference costs. Moreover, adapters are used allowing us to tune smaller number of parameters (~ 10 million) resulting in huge training cost savings. Our model is also benchmarked on 13 public datasets and outperforms strong baselines like Leopard. Current work is underway to make our training dataset more diverse so that our models are more generalizable.

Ethical Considerations

Due to various legal and institutional concerns arising from the sensitivity of clinical data, it is difficult for researchers to gain access to relevant data except for MIMIC (Johnson et al., 2016). Despite its large size (covering over 58k hospital admissions), it is only representative of patients from a specific clinical domain (the intensive care unit) and geographic location (a single hospital in the United States). We can not expect such a sample to be representative of either the larger population of patient admissions or other geographical regions/hospital systems. We have tried to address this partially by collecting radiology data for various body parts across multiple practices in the US. However we are always mindful that our work may not generalize to new body parts/pathologies and radiology practices (see Section H). Even though we introduce a simple OOD metric, we realize it is far from perfect. We understand the need to minimize ethical risks of AI implementation like threats to privacy and confidentiality, informed consent, and patient autonomy. And thus we strongly believe that stakeholders should be flexible in incorporating AI technology as a complementary tool and not a replacement for a physician. Thus, we develop our workflows, annotation guidelines and generate actionable insights by working in conjunction with a varied group of radiologists and medical professionals to minimize these above risks. Finally our pipeline as deployed is meant as a pseudo-labeling tool which we expect would cut down on expensive annotation costs but can potentially introduce some bias in our pseudo-labels.

References

- Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. 2019. [Infinite mixture prototypes for few-shot learning](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 232–241. PMLR.
- Emily Alsentzer, John Murphy, William Boag, Weihung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. [Publicly available clinical BERT embeddings](#). In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.
- Jason Altschuler, Sinho Chewi, Patrik Robert Gerber, and Austin J Stromme. 2021. [Averaging on the bures-wasserstein manifold: dimension-free convergence of gradient descent](#). In *Advances in Neural Information Processing Systems*.
- Trapit Bansal, Rishikesh Jha, and Andrew McCallum. 2020a. [Learning to few-shot learn across diverse natural language classification tasks](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5108–5123, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020b. [Self-supervised meta-learning for few-shot natural language classification tasks](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534.
- Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1997. [On the optimization of a synaptic learning rule](#).
- 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.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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](#). In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.
- Marco Cuturi and Arnaud Doucet. 2014. [Fast computation of wasserstein barycenters](#). In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 685–693, Beijing, China. 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.
- P Dimond, Paul Fadale, Michael Hulstyn, Glenn Tung, and J Greisberg. 1998. [A comparison of mri findings in patients with acute and chronic acl tears](#). *The American journal of knee surgery*, 11:153–9.

- Ning Ding, Yulin Chen, Xiaobin Wang, Hai-Tao Zheng, Zhiyuan Liu, and Pengjun Xie. 2022. [Few-shot learning with big prototypes](#).
- Kawin Ethayarajh. 2019. [How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings](#). 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 55–65, Hong Kong, China. Association for Computational Linguistics.
- Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. [Model-agnostic meta-learning for fast adaptation of deep networks](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.
- Yao Ge, Yuting Guo, Yuan-Chi Yang, Mohammed Ali Al-Garadi, and Abeed Sarker. 2022. [Few-shot learning for medical text: A systematic review](#).
- Ruiying Geng, Binhua Li, Yongbin Li, Xiaodan Zhu, Ping Jian, and Jian Sun. 2019. [Induction networks for few-shot text classification](#).
- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. [Domain-specific language model pretraining for biomedical natural language processing](#). *ACM Trans. Comput. Healthcare*, 3(1).
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. [Towards a unified view of parameter-efficient transfer learning](#). In *International Conference on Learning Representations*.
- Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. [spaCy: Industrial-strength Natural Language Processing in Python](#).
- Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. [Meta-learning in neural networks: A survey](#).
- 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):160035.
- Rabeeh Karimi Mahabadi, Sebastian Ruder, Mostafa Dehghani, and James Henderson. 2021. [Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks](#). In *Annual Meeting of the Association for Computational Linguistics*.
- Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015a. [Human-level concept learning through probabilistic program induction](#). *Science*, 350(6266):1332–1338.
- Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015b. [Human-level concept learning through probabilistic program induction](#). *Science*, 350(6266):1332–1338.
- Arjun Nair, Anand Devaraj, Matthew E J Callister, and David R Baldwin. 2018. [The fleischner society 2017 and british thoracic society 2015 guidelines for managing pulmonary nodules: keep calm and carry on](#). *Thorax*, 73(9):806–812.
- Alex Nichol, Joshua Achiam, and John Schulman. 2018. [On first-order meta-learning algorithms](#).
- 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.
- Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. 2020. [AdapterHub: A framework for adapting transformers](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.
- Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. 2019. [Rapid learning or feature reuse? towards understanding the effectiveness of maml](#).
- Rebecca Roelofs, Vaishaal Shankar, Benjamin Recht, Sara Fridovich-Keil, Moritz Hardt, John Miller, and Ludwig Schmidt. 2019. [A meta-analysis of overfitting in machine learning](#). In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Jürgen Schmidhuber. 1987. [Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook](#). Diploma thesis, Technische Universität München, Germany, 14 May.
- Arijit Sehanobish, McCullen Sandora, Nabila Abraham, Jayashri Pawar, Danielle Torres, Anasuya Das, Murray Becker, Richard Herzog, Benjamin Odry, and Ron Vianu. 2022. [Explaining the effectiveness of multi-task learning for efficient knowledge extraction from spine MRI reports](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 130–140, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

- Rishav Singh, Vandana Bharti, Vishal Purohit, Abhinav Kumar, Amit Kumar Singh, and Sanjay Kumar Singh. 2021. [Metamed: Few-shot medical image classification using gradient-based meta-learning](#). *Pattern Recognition*, 120:108111.
- Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17*, page 4080–4090, Red Hook, NY, USA. Curran Associates Inc.
- Erik E. Swartz, R. T. Floyd, and Mike Cendoma. 2005. [Cervical Spine Functional Anatomy and the Biomechanics of Injury due to Compressive Loading](#). *Journal of athletic training*, 40(3):155–161.
- Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. 2016. [Matching networks for one shot learning](#). In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.
- C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. 2011. Caltech-ucsd birds-200-2011. Technical Report CNS-TR-2011-001, California Institute of Technology.
- Jixuan Wang, Kuan-Chieh Wang, Frank Rudzicz, and Michael Brudno. 2021. [Grad2task: Improved few-shot text classification using gradients for task representation](#). In *Advances in Neural Information Processing Systems*.
- 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.
- Dani Yogatama, Cyprien de Masson d’Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. 2019. [Learning and evaluating general linguistic intelligence](#).
- Jian Zhang, Chenglong Zhao, Bingbing Ni, Minghao Xu, and Xiaokang Yang. 2019. [Variational few-shot learning](#). In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1685–1694.

A Annotation

First we collect data from various sources and a part of the data are annotated by our team of in-house expert annotators with deep clinical expertise, which we use as test and development sets for

our model training. We then use this annotated data to train a larger pool of other annotators who are generally medical students. They are provided clear guidelines on the task and performance is measured periodically on a benchmark set and feedback is provided. As of the writing of the manuscript, the validation and the test sets as described in section 3 are being used to train the annotators. After the completion of their training, the annotators will annotate the remaining unlabeled data that will be used as a training data for our models. The entire process is slow but is designed to generate high quality annotated data. We believe that our few shot models can be used as a source of pseudo-labels and will greatly simplify and quicken our annotation process.

B Shoulder Dataset

In this section we will briefly describe our label rich shoulder dataset that is used as meta-training and meta-validation sets. There are 80 labels for the shoulder dataset. They range from Clinical history, metadata, Impressions, Finding to various granular pathologies at different structures in the shoulder like AC joint, Rotator Cuff, Muscles, Bursal Fluid, Supraspinatus, Infraspinatus, Subscapularis, Labrum, Glenohumeral Joint, Humeral Head, Acromial Morphology, Impingement: AC Joint. The labels are split such that all pathologies in a given structure appear at either training or validation but not both. We believe that such a split would help a model to learn the key words that may describe the granularity of a pathology in a given structure of interest. The dataset level statistics can be found in figure 1 and table 1. An example of the shoulder data is shown in figure 4.

Text	Labels
The AC joint and anterior acromion show evidence of prior subacromial decompression and there may have been a distal clavicle excision as well with widening of the AC fluid in the joint glenohumeral joint/ labrum.	AC Joint: Mild Arthritis with Edema
Type II acromion with hypertrophic changes causing impingement and partial rotator cuff tear of the infraspinatus and supraspinatus myotendinous junction.	Impingement: Acromion
Mild subacromial-subdeltoid bursitis. Findings are age-indeterminate unless otherwise specified.	Bursal Fluid: Small
Acromioclavicular joint: Anatomic alignment. No substantial degenerative change.	AC Joint: Normal
There is fraying of the anterior labrum above the level of the equator.	Labrum: Normal or mild degeneration

Figure 4: Figure showing an example of our shoulder dataset which is used for meta-training. Note that the labels attached to the text have information about the location and severity of a given pathology.

C Detailed Workflow

We now present a detailed description of various body part specific workflows. All reports, irrespective of body part, are first de-identified according to HIPAA regulations. We then pass the report through a sentence parser to parse the report in sentences.

C.1 Lung Dataset

For the lung dataset, we use a report segmenter which is a rule-based regex to extract the "Impression" section from the entire report. This section can be thought as the summary of the report and contains all the critical information like number of lung nodules and their sizes and potential for malignancy. This section text is used for final classification task as shown in figure 5. Figure 6 shows examples of the labels in the dataset.

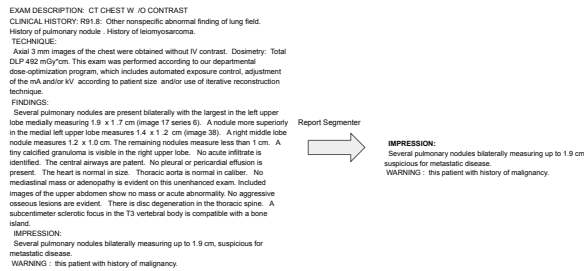


Figure 5: Figure showing the preprocessing of the lung dataset. Our report segmenter extracts the relevant paragraph which is used for downstream classification.

Text	Labels
Dominant 6.7 cm right upper lobe mass, with contiguous extension into the right hilar region, probable adjacent interstitial spread, bilateral pulmonary metastatic nodules, mediastinal lymphadenopathy. 3 cm left adrenal gland lesion suspicious for metastasis.	High Risk
Mild interstitial and ground glass density in the right upper lobe near the apex and probably representing sequelae of radiation treatment. A tiny benign-appearing pulmonary nodule in the right middle lobe is unchanged compared to 2016. No findings suspicious for metastatic disease in the chest. Hepatic steatosis.	Low Risk

Figure 6: Figure showing the labels in the Lung dataset

C.2 Cervical Dataset

Our task in the cervical dataset is to predict the severity of a neural foraminal stenosis for each motion segment - the smallest physiological motion unit of the spinal cord (Swartz et al., 2005). Breaking information down at the motion segment level in this way enables pathological findings to be correlated with clinical exam findings, and can inform future treatment interventions. A BERT based NER model is used to identify the motion segment(s) referenced in each sentence, and all the sentences containing a particular motion segment

are concatenated together. We also use additional rule-based logic to assign motion segments to relevant sentences that may not mention a motion segment in it. We then predict the disease severity using this concatenated text at *each* motion segment. This data pre-processing mostly follows the ideas and the steps outlined in (Sehanobish et al., 2022). Figure 7 shows our preprocessing steps and figure 8 shows examples in the datasets.

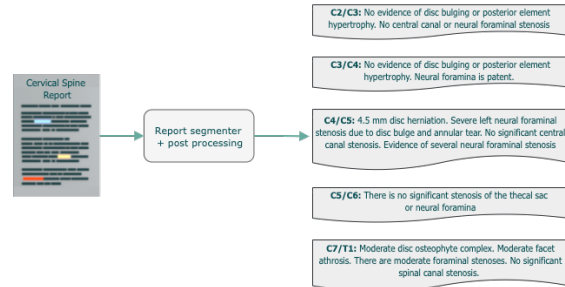


Figure 7: Figure showing the preprocessing of the cervical dataset. Our report segmenter extracts all the motion segments mentioned in the report and groups all sentences belonging to the relevant motion segment. The paragraph belonging to a given motion segment is used for downstream classification.

Text	Label
C3-4: Desiccation mild disc space loss. There is mild disc osteophyte bulge. Uncovertebral hypertrophy left greater than right. Mild facet hypertrophy. In conjunction with short pedicles mild central canal narrowing. There is mild to moderate left and mild right foraminal stenosis.	0
At C3-C4, there is a degenerated bulging disc with osteophytic ridging, facet and uncovertebral hypertrophy and short pedicles combining to cause mild to moderate central stenosis and severe bilateral foraminal stenosis unchanged. There is mild motion artifact on some of the images. There is reversal of the cervical alignment with grade 1 anterolisthesis at C3-4 and C5-6.	1

Figure 8: Figure showing the labels in the Cervical dataset. 0 means absence of severe neural foraminal stenosis and 1 indicates presence of severe neural foraminal stenosis.

C.3 Knee Dataset

The data processing steps for the knee dataset is similar to the cervical dataset. A BERT based NER model is used to tag sentences that mention the structure of importance, i.e. the anterior cruciate ligament (ACL). We group all the sentences together that mention ACL and we use these grouped sentences to predict our pathology severity as shown in the workflow (figure 9). An example of the labels in the knee dataset can be found in figure 10.

D Additional Experiments

We also experiment with BERT-base and Clinical BERT as additional backbones. We add adapters

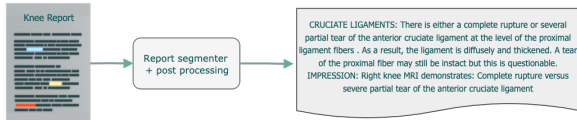


Figure 9: Figure showing the preprocessing of the knee dataset. Our report segmenter selects all the relevant sentences pertaining to the structure of interest, i.e. ACL. We then predict various pathology severities using this paragraph of text.

Text	Acute Tear	Complete Tear
The anterior cruciate ligament is intact and there is a partial tear of the posterior cruciate ligament with a thin residual component of the distal half of the PCL still intact.	0	0
While there is some edema in the ACL, there appear to be intact fibers. This may indicate ACL sprain or partial tear. A complete tear is not identified. Secondary signs of ACL insufficiency are not identified. While there is likely an ACL sprain or mild partial tear, there are intact ACL fibers. The pattern of bone bruises raises some concern for an ACL tear, but the femoral bone bruises slightly more lateral than commonly seen.	1	0
Proximal ACL tear and PCL intact. Left knee MRI demonstrates: Complete ACL tear with bone bruises in the medial tibial plateau, medial femoral condyle and lateral femoral condyle.	1	1
There is no normal anterior cruciate ligament identified. There is diffuse intermediate signal within the posterior cruciate ligament on the proton density images and to a lesser extent on the T2-weighted images compatible with chronic PCL degeneration. While this could be the sequela of old surgery, degenerative tearing of the meniscus including involvement of the root attachment cannot be excluded. Nonvisualization of the ACL compatible with an old ACL tear.	0	1

Figure 10: Figure showing the labels in the Knee dataset. 0 means absence of a given pathology and 1 indicates presence of such.

to these backbones as well. Finally, we choose the best model based on meta-validation accuracy and use it for our downstream tasks. In all our experiments, PubMedBERT-based backbones outperform the BERT-base and the Clinical BERT backbones.

E Hyperparameters and Additional Experimental Details

In this section, we will describe the hyperparameters used for experiments on our internal and public datasets and explain some of the design choices. Table 8 shows the best hyperparameters used for our experiments. For our internal dataset, we use the Pfeiffer configuration in the adapter implementation from (Pfeiffer et al., 2020), whereas for the public datasets we use the exact implementation and configuration as in (Wang et al., 2021) for a fair comparison to the results reported there. For all vanilla ProtoNet experiments, we use the Euclidean distance as it outperforms the cosine distance. All BERT models without adapters are trained with 8 shots and 8 support due to memory considerations. We choose learning rate and the variance regularizer for each model from $\{1e-5, 2e-5, 5e-5, 1e-4\}$ and $\{1e-4, 1e-3, .01, .1, .5\}$ based on the validation performance. For all the experiments, a dropout

Backbone	Methods	Accuracy
BERT-base	Vanilla ProtoNet	86.3 ± 1.2
	Big ProtoNet	87.8 ± .9
	Leopard	81.4 ± 9.7
	ProtoNet w/ Isotropic Gaussian	88.7 ± 1.4
	ProtoNet w/ Isotropic Gaussian + reg	89.5 ± .8
BERT-base w/Adapters	Variance Aware ProtoNet (ours)	88.9 ± 1.5
	Variance Aware ProtoNet + reg (ours)	90.1 ± .9
	Vanilla ProtoNet	85.6 ± 1.3
Clinical BERT	Big ProtoNet	87.1 ± 1.1
	Leopard	87.8 ± .8
	ProtoNet w/ Isotropic Gaussian	88.6 ± .7
	ProtoNet w/ Isotropic Gaussian + reg	88.1 ± 1.2
	Variance Aware ProtoNet (ours)	89.7 ± .8
Clinical BERT w/Adapters	Variance Aware ProtoNet + reg (ours)	87.4 ± 1.3
	Big ProtoNet	88.5 ± 1.1
	Leopard	82.2 ± 9.8
	ProtoNet w/ Isotropic Gaussian	89.6 ± 1.2
	ProtoNet w/ Isotropic Gaussian + reg	90.1 ± .8
Clinical BERT w/Adapters	Variance Aware ProtoNet (ours)	89.9 ± 1.1
	Variance Aware ProtoNet + reg (ours)	90.9 ± .8
	Vanilla ProtoNet	86.8 ± .9
	Big ProtoNet	87.9 ± 1.1
	ProtoNet w/ Isotropic Gaussian	88.4 ± 1.3
Clinical BERT w/Adapters	ProtoNet w/ Isotropic Gaussian + reg	89.1 ± .9
	Variance Aware ProtoNet (ours)	88.7 ± 1.1
	Variance Aware ProtoNet + reg (ours)	89.5 ± .9

Table 6: Results showing accuracy percentages on the meta-validation dataset. We sample 1000 tasks with 4-way 8-shot and 16-support classification. We replicate each experiment over 10 random seeds.

layer is added after the final BERT layer.

For our internal dataset, we also experiment with $\{2, 3, 4\}$ ways and $\{4, 6, 8\}$ shots and $\{4, 6, 8, 12, 16\}$ support. The experiments with 2-way and 3-way produce poor results on our downstream tasks irrespective of the number of shots and support. During training with 4-way, the meta-validation results for lower support show worse performance than the numbers reported in table 4. We believe that it is caused by the high variability between the various groups of samples of a given class. Finally our downstream performance is best for models that are trained on higher number of shots.

In case of ProtoNets, there is no adaptation during testing. The validation set is used to compute prototypes to query the test set. However, in case of Leopard, there is an additional few shot adaptation step that occurs as outlined in (Wang et al., 2021). In this case, the validation set is used for the adaptation and also as the support set for querying the test set.

E.1 Effect of Regularization on means and variances

Table 9 illustrates the benefits of adding the regularization term. The regularization term not only aids in lowering the variances but also manages

Backbone	Methods	Foraminal	Knee (Acute Tear vs Not)	Knee (Complete tear vs Not)	Lung
BERT-base	Baseline	.24	.29	.32	.19
	Multi-Task	.29	.34	.41	.27
	Vanilla ProtoNet	.75	.71	.66	.65
	Big ProtoNet	.57	.58	.53	.6
	Leopard	.63	.72	.61	.41
	ProtoNet w/ Isotropic Gaussian	.77	.72	.69	.68
	ProtoNet w/ Isotropic Gaussian + reg	.78	.76	.71	.70
	Variance Aware ProtoNet (ours)	.79	.78	.73	.72
	Variance Aware ProtoNet + reg (ours)	.81	.80	.76	.75
BERT-base w/ Adapters	Baseline	.28	.32	.40	.25
	Multi-Task	.32	.35	.44	.29
	Vanilla ProtoNet	.74	.73	.65	.67
	Big ProtoNet	.58	.59	.55	.61
	ProtoNet w/ Isotropic Gaussian	.78	.71	.67	.69
	ProtoNet w/ Isotropic Gaussian + reg	.80	.74	.72	.74
	Variance Aware ProtoNet (ours)	.80	.74	.72	.74
	Variance Aware ProtoNet + reg (ours)	.82	.77	.77	.78
	Clinical BERT	Baseline	.31	.37	.42
Multi-Task		.34	.45	.47	.38
Vanilla ProtoNet		.77	.72	.68	.66
Big ProtoNet		.57	.59	.53	.61
Leopard		.74	.78	.77	.62
ProtoNet w/ Isotropic Gaussian		.78	.74	.71	.68
ProtoNet w/ Isotropic Gaussian + reg		.80	.76	.74	.71
Variance Aware ProtoNet (ours)		.82	.79	.76	.74
Variance Aware ProtoNet + reg (ours)		.84	.81	.79	.76
Clinical BERT w/ Adapters	Baseline	.35	.42	.45	.33
	Multi-Task	.37	.45	.49	.37
	Vanilla ProtoNet	.76	.74	.70	.67
	Big ProtoNet	.58	.60	.57	.62
	ProtoNet w/ Isotropic Gaussian	.79	.76	.72	.70
	ProtoNet w/ Isotropic Gaussian + reg	.81	.77	.73	.72
	Variance Aware ProtoNet (ours)	.83	.81	.76	.73
	Variance Aware ProtoNet + reg (ours)	.85	.82	.81	.77

Table 7: Table showing F1 scores of few shot models with BERT-base and Clinical BERT backbones in downstream classification tasks.

to push the centroids away further which we believe sheds some light on our method’s success in downstream classification tasks. We also carry out various ablation studies by changing the regularization hyperparameter.

E.2 Metric and other modeling choices

In (Snell et al., 2017), only the sample means (i.e. means of the support vectors) are used to estimate the true population mean. In fact, by the Central limit theorem, we can use the sample variance (after normalization) to get an unbiased estimate of the population variance. Unlike the original work, we sought to use this extra information to better understand the class distribution. The Gaussian assumption is strong but it is motivated by the fact that it allows us to compute Wasserstein distances in a computationally tractable manner. Finally to motivate the choice of using the Wasserstein distance instead of a Bergman divergence like KL diver-

gence, consider the following motivating example, $N_1(\mu_1, \Sigma_1), N_2(\mu_2, \Sigma_2)$ be 2 Gaussians and for simplicity assume: $\Sigma_1 = \Sigma_2 = wI$ and $\mu_1 \neq \mu_2$. With these assumptions, $W^2 = \|\mu_1 - \mu_2\|^2$ And $D_{KL} = \frac{1}{2w}\|\mu_1 - \mu_2\|^2$. Note that Wasserstein distance does not change if the variance changes (w can be arbitrarily large) whereas the KL divergence does. In fact, this is pointed out in (Ding et al., 2022) where their goal is to create spherical Gaussians with large radii. However, we found that having large variance produces worse results in our downstream tasks. Finally similar dependence on variance is in play if one computes a simple likelihood of the sample in the class distributions.

F Public Benchmarks

In this section, we describe the training procedure for the public benchmark datasets. The baseline results are taken from (Wang et al., 2021;

Algorithm 1: PyTorch style Pseudocode for Variance Aware ProtoNets

```

/* f: Encoder Network */
/* N: dimension of the representation */
/* c: Number of classes or ways */
/* k: Shots or number of examples per class in the query set */
/* m: Supports per class in the support set */
/* dist: Pairwise squared Euclidean distance function */
/* loss_fn: Cross-Entropy Loss function */
/* λ: regularizer */
Input: Sample a set  $L$  of labels, mini-batch of Support set  $S_L$ , Query set  $Q_L$ 
/* Compute statistics for each class in the Support set */
sorted_labels = torch.sort(support_labels) // sort the labels in the support set
c = len(s.values.unique()) // Number of ways
support_sorted = support[sort.indices]
labels_sorted = labels[sort.indices]
embeddings_support = f(support_sorted) // m * c * N
m = embeddings_support.shape[0] // support per class
embeddings_support = embeddings_support.reshape(c, m, -1) // c * m * N
support_mean = embeddings_support.mean(dim=1) // c * N
support_var = torch.var(embeddings_support, dim=1)**2 // c * N
/* Get embeddings for the query set and compute distances from the support */
query = f(Q_L) // k * N
logits = dist(query, support_mean) + torch.sum(support_var, dim=1) // k * c, adding trace to the
distance matrix
loss = loss_fn(-logits, query_labels) + λ * (torch.norm(support_var, dim=1))/c // Regularizer term

```

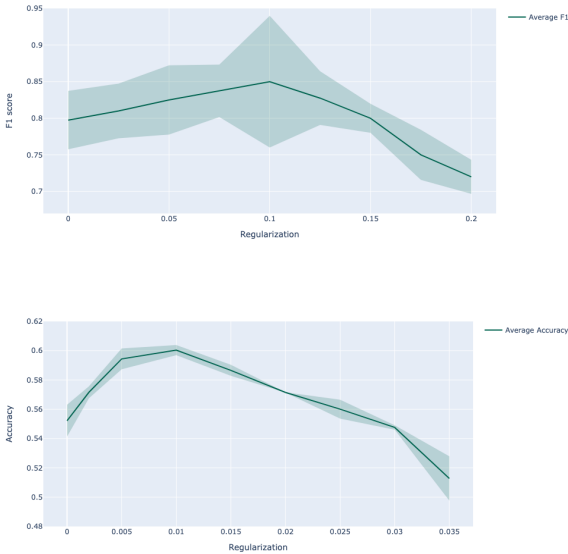


Figure 11: Figure showing the effect of changing the regularization hyperparameter. Top: Figure showing the F1 score averaged over our 4 internal datasets. Bottom: Figure showing 16-shot accuracy averaged over 13 public datasets. We see a similar trend for 4 and 8-shot accuracies for these public datasets as well.

Hyperparameter Type	Internal Dataset	Public Datasets
Epochs	30	40
Sequence Length	128	128
Optimizer	AdamW	AdamW
Learning Rate	$3e - 5$	$2e - 5$
Weight Decay	$1e - 4$	$1e - 4$
Gradient Clip	3	2
Early Stopping	Yes	Yes
Learning Rate Scheduler	Linear	Linear
Dropout	.1	.1
Shots	8	8
Number of supports	16	16
Variance Regularizer	.1	.01

Table 8: Hyperparameters used for all our Variance Aware ProtoNet experiments with BERT+Adapter backbones

Bansal et al., 2020a). We have followed the same meta-training procedure as described in (Wang et al., 2021). Specifically, for meta-training, WNLI (m/mm), SST-2, QQP, RTE, MRPC, QNLI, and the SNLI datasets (Bowman et al., 2015) are used. The validation set of each dataset is used for hyperparameter searching and model selection. The models are trained by sampling episodes from the meta-training tasks. The sampling process first selects a

Experiment type	Without regularization			With regularization		
Task Names	Distance	Norm of	Norm of	Distance	Norm of	Norm of
	between centroids	class 0 variance	class 1 variance	between centroids	class 0 variance	class 1 variance
Lung	2.97	1.12	1.31	3.96	1.15	1.08
Foraminal	3.65	1.71	1.59	4.17	1.62	1.38
Knee (ACL complete tear)	4.12	1.97	2.10	5.01	1.87	1.85
Knee (ACL acute tear)	3.95	1.53	1.49	4.32	1.27	1.35

Table 9: Table showing the showing the class statistics with and without regularization. Higher Distance and Lower variance is better.

dataset and then randomly selects m examples for each class as the support set and another k -shots as the query set and the probability of a selected task is proportional to the square root of its dataset size (Bansal et al., 2020b). For meta-testing, we use 13 datasets ranging from NLI, text classification and sentiment analysis. For the models and datasets marked with *, we use the results reported in (Bansal et al., 2020a) and for those datasets, we use the code from (Wang et al., 2021) to generate the results for ProtoNet with Bottleneck Adapters while the rest of the results are taken from (Wang et al., 2021). We reuse their implementation and configuration of their adapters but modify the loss function with the Wasserstein distance along with our variance regularization term. Table 5 shows the superior performance of our method beating all the baselines. For detailed hyperparameters, please see section E. Our method without the variance regularization term shows similar performance to that of the Leopard baselines. For the isotropic variant method, it shows similar performance to Leopard with the variance regularization term and worse without.

G Stability of the Prototypes

For simplicity, we use our entire validation sets to compute prototypes. In this section we show how our results vary if we choose a subset of our validation set to create the prototypes. The figure 12 shows the F1 scores when a subset of the data is used to compute the prototypes and the variances for a given class.

H Failure Cases

We also test our models on few additional tasks like (i) predicting the severity of disc herniation in our cervical dataset and (ii) predict the presence of cord compression at various motion segments in our internal dataset on the lumbar spine. Our models achieve an F1 score of .51 and .39 respectively. The figure 13 shows how the classes are distributed.

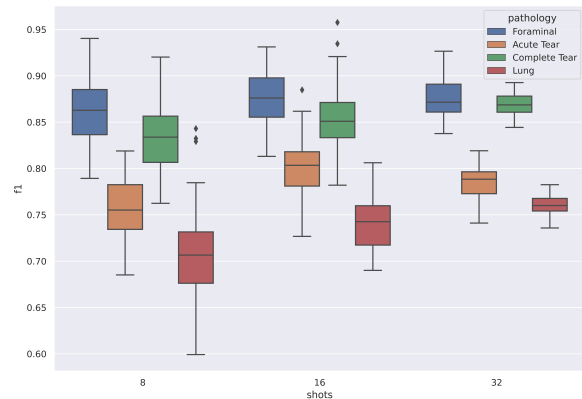


Figure 12: Figure showing stability of the prototypes. We sample k examples 50 times to construct the prototypes and the standard deviations.

We attribute the failures to the poor separability between classes and the high variance in the data distribution.

It is an ongoing project to understand what makes our model work for these downstream tasks and why our model works on some tasks and not others. We hope that by simply increasing the diversity of our training data or applying newer adapter architectures like Mix-and-Match Adapter (He et al., 2022) and Compacter (Karimi Mahabadi et al., 2021), our current methods will work on a wide range of downstream pathologies.

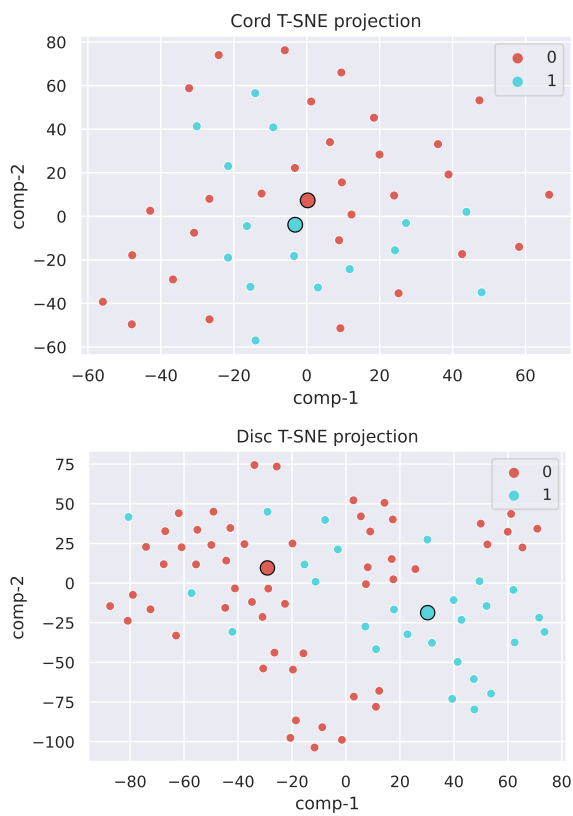


Figure 13: T-Sne projections of our Cord and Disc Data. The prototypes for cord classes are very close while the prototypes for disc are well separated. However the large variance in the disc classes causes bad performance.

Named Entity Recognition in Industrial Tables using Tabular Language Models

Aneta Koleva^{1,2*}, Martin Ringsquandl^{1*}, Mark Buckley¹, Rakebul Hasan¹ and Volker Tresp^{1,2}

¹Siemens AG, ²Ludwig-Maximilians University

first_name.last_name@siemens.com

Abstract

Specialized transformer-based models for encoding tabular data have gained interest in academia. Although tabular data is omnipresent in industry, applications of table transformers are still missing. In this paper, we study how these models can be applied to an industrial Named Entity Recognition (NER) problem where the entities are mentioned in tabular-structured spreadsheets. The highly technical nature of spreadsheets as well as the lack of labeled data present major challenges for fine-tuning transformer-based models. Therefore, we develop a dedicated table data augmentation strategy based on available domain-specific knowledge graphs.

We show that this boosts performance in our low-resource scenario considerably. Further, we investigate the benefits of tabular structure as inductive bias compared to tables as linearized sequences. Our experiments confirm that a table transformer outperforms other baselines and that its tabular inductive bias is vital for convergence of transformer-based models.

1 Introduction

There has been growing interest in developing special model designs intended to capture tabular structure (Deng et al., 2020; Yin et al., 2020; Herzig et al., 2020; Wang et al., 2021). A recent survey named these models tabular language models (TaLMs) and provided an overview of the different architectures and pretraining objectives (Dong et al., 2022). One of the downstream tasks where TaLMs are applicable is table interpretation (TI) with its sub-tasks: entity linking, column type annotation and relation extraction (Deng et al., 2020). Most TaLMs for TI use BERT as the backbone language model (LM) for encoding the content of table cells and aggregate their representations on

different levels (cell, row or column) depending on the task.

Although tabular data is omnipresent in industry, TaLMs such as table transformers, have not found their way into industrial applications yet. One reason being the nature of data stored in industrial tables which is different and more dynamic than data in academic datasets where the schema of the table is consistent and each cell contains a single entity of one type (Cutrona et al., 2020). As shown in Figure 1, industrial tables contain multiple *sub-cell* entities from different types, hence the TaLMs which provide cell-level aggregation are not sufficient. In this direction, we formulate the problem of sub-cell named entity recognition (NER) in tables using TaLMs.

Another challenge is that tabular data in industry is often lacking labels, especially labels reflecting the high variance across examples. Due to the very technical and domain-specific nature only experts can effectively provide such labels, which is – for most tasks – too expensive. These low-resource scenarios are challenging for statistical NLP models and usually prohibit fine-tuning of large-scale transformer-based models. A popular strategy to remedy low-resource scenarios is data augmentation (DA) (Simard et al., 1996), which allows to increase data diversity without having to collect new examples. Common DA techniques in NLP range from using external knowledge such as WordNet (Zhang et al., 2015), machine-translation models for back-translation (Yaseen and Langer, 2021) or mixing of examples inspired from computer vision (Yun et al., 2019). An empirical study by (Longpre et al., 2020) showed that applying off-the-shelf DA techniques (Sennrich et al., 2016; Wei and Zou, 2019) for fine-tuning of LM like BERT or RoBERTa bring little to no improvement and become even less beneficial in cross-domain settings (Herzig et al., 2020; Zhong et al., 2020). These studies emphasize the challenge of

* Equal Contribution.

No.	Eq.name	Risk rating	min/max nominal pressure	Pow rating (kW)	Ow. Code
P-47-01	Area 47 - Cent. pump for refinement	High	20/30 bar Pump 10/20 bar Pipe	20	5MW
T-41-02/A , T-41-02/B	Wtr Tank (qty: 2)	Low	Inner: 10/20 Outer: 5/10	- Capacity : 500 l	2AV

QUANT
UoM
TAG
EQ

Figure 1: Example table from an industrial plant equipment spreadsheet. Boxes represent NER annotations.

developing domain-specific DA techniques which would help improve the existing pretrained transformer models.

Although, there are no domain-specific DA techniques applicable to a tabular dataset, in many industrial domains there exist external resources which can be exploited for creating augmented tables. In this paper we study a DA technique for industrial spreadsheet tables leveraging publicly available resource based on an industrial standard. Specifically, the contributions of this paper are the following:

- We introduce a table transformer model for sub-cell NER, TABNER, and provide an industrial use case as a motivation for this. To the best of our knowledge, this is the first attempt to solve NER in tables with TaLMs.
- We develop a novel DA technique for semantically consistent augmentation of tables based on domain-specific knowledge graphs.
- We empirically show that the inductive bias of TaLMs is valuable and combined with our DA technique boosts the performance by 9% compared to a sequential model.

2 Industry NER Use Case

As motivation for tabular NER in an industrial context, we describe a real-world dataset from which information about industrial plant equipment, such as actuators, sensors, vessels, etc. and their physical quantities should be extracted. This information is typically collected and maintained by engineers in spreadsheets. The spreadsheets are roughly organized in a tabular format, as can be seen from the example table in Figure 1. In these spreadsheets, each row typically represents information about one or multiple equipment instances. Some columns represent relevant physical properties of these equipments, while others are non-informative. However,

Dataset	μ_{tok}	σ_{tok}	K_{tok}	μ_{col}	σ_{col}
SemTab	2	2.5	132.2	4.5	1.9
Plant	2.6	3.7	585.3	16.3	21.6

Table 1: Dataset statistics: academic vs. industry.

the engineers do neither comply to a fixed schema nor to unified spelling of equipment or properties. The goal is to automatically extract relevant entities for creating a structured specifications of the plant equipment. We phrase this problem as NER task with the following types of entities. The type *TAG* refers to a systematic identifier of an equipment. There are some conventions for generating equipment tags (e.g. NOROK, KKS standards), but most plant operators customize them and some sheets do not contain identifiers at all. Type *EQ* is for surface names of equipment types. The type *QUANT* refers to the physical properties/quantities describing the functional specifications of equipment and the type *UoM* stands for unit of measurement.

Table Statistics It is not obvious why performing NER in tables would benefit from sophisticated language models. In fact, looking at common tabular benchmark datasets, such as the ones used in the SemTab challenge (Cutrona et al., 2020), detecting entities is usually very straightforward. Since all tokens in a cell are assumed to represent a single entity, sub-cell NER is an unnecessary step and we only need to perform entity/cell linking. Looking at the example table in Figure 1, however, gives the impression that these industrial spreadsheets are very differently structured from common benchmarks. There can be quite some text and even multiple sub-cell entities in a single cell. Table 1 supports this impression with statistical evidence. The average number of tokens per cell, μ_{tok} , is 30% higher in our industrial dataset compared to a dataset from SemTab. Further, its standard devia-

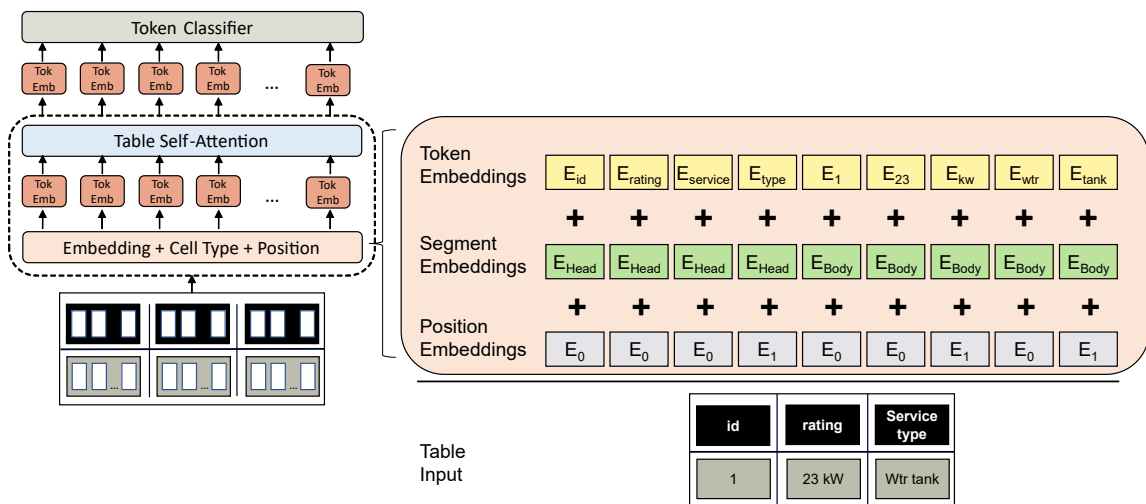


Figure 2: Input modifications to vanilla transformer to encode tokens with tabular structure.

tion σ_{tok} and the Kurtosis K_{tok} , show that there is more variance due to the much longer tail of the distribution of number of tokens in the plant tables. Even more obvious is the difference at the column level where the tables in the SemTab challenge contain on average 4 times less columns (μ_{col}) than the tables describing plant equipment specification with much lower variance as well. This suggests that every token in our NER task has a much broader intra- and inter-cell context.

3 Related Work

There has been some research focused on extracting entities and their quantities from web tables. Ibrahim et al. (Ibrahim et al., 2016) phrased this problem as entity linking using a table-biased Markov random field and distant supervision.

Wu et al. (Wu et al., 2018) employed BiLSTM models to encode rich-format documents (unstructured text, headings, tables) that mention electronic components, quantities and units of measure. They used hand-crafted labeling functions for collecting (weakly) labeled entities and relations which can be used as weak supervision.

A recent work on table classification (Koleva et al., 2021) compared TaLMs like TaBERT (Yin et al., 2020) versus non-contextual word embedding methods for generating table vector representations. TURL (Deng et al., 2020) uses a Transformer (Vaswani et al., 2017) with table-specific attention mechanism which has been pre-trained and fine-tuned towards solving the tasks of table interpretation: column type annotation, entity linking and relation extraction. However, this methods gen-

erates representations on a cell level and therefore can not be applied for solving our NER problem.

We are not aware of any work that uses TaLMs for sub-cell table NER in an industrial setting.

Data Augmentation Recently, many different DA techniques have been proposed with the purpose to solve low-resource issues in NLP by generating new examples from existing datasets. For a comprehensive overview on the different DA techniques, we refer the readers to the recent survey by Feng et al. (Feng et al., 2021).

Several simple and effective DA techniques for NER are presented by (Dai and Adel, 2020). However, these techniques are not directly applicable to the industrial tabular data since they rely on domain-agnostic linguistic resources like WordNet. Similarly, methods for sequence labeling, such as backtranslation (Yaseen and Langer, 2021) can not be applied to tabular data because the content of the tables are mostly facts and not full sentences.

4 Method

We now define the table NER problem and outline how we encode tokens in tables using TaLMs.

We define a table as a tuple $T = (C, H)$, where $C = \{c_{1,1}, c_{1,2}, \dots, c_{i,j}, \dots, c_{n,m}\}$ is the set of table body cells for n rows and m columns. Every cell $c_{i,j} = (w_{c_{i,j},1}, w_{c_{i,j},2}, \dots, w_{c_{i,j},t})$ is a sequence of tokens of length t . The table header $H = \{h_1, h_2, \dots, h_m\}$ is the set of corresponding m column header cells, where $h_j = (w_{h_j,1}, w_{h_j,2}, \dots, w_{h_j,q})$ is a sequence of header tokens with length q . We use $T_{[i,:]}$ to refer to the i -th row ($H = T_{[0,:]}$) and $T_{[:,j]} =$

$\{h_j, c_{1,j}, \dots, c_{n,j}\}$ to refer to the j -th column of T .

Each labeled cell has an NER-tag sequence: $(y_1, y_2, \dots, y_{|cell|})$, where each $y_i \in \mathcal{Y}$. We use IO tags, thus \mathcal{Y} is $\{O\} \cup \{I-ENT\}$, where $ENT \in \{TAG, EQ, QUANT, UoM\}$.

4.1 TABNER Model

Compared to the existing TaLMs such as TaBERT (Yin et al., 2020), TURL or TAPAS (Herzig et al., 2020) which generate cell-level representations, we propose a simple modification to the vanilla transformer (Vaswani et al., 2017) which allows us to use almost any pre-trained transformer¹ to obtain a (sub-cell) token-level representations for a table.

Our TABNER model consists of a token encoder layer ENC and a classification layer. A conceptual architecture of the table token input encoding is shown in Figure 2, where token vector representations for each token in the linearized table are generated by aggregating the token embeddings, the segment embeddings, and position embeddings. The segment indicates if a token is part of the head or the body (instead of the 1st / 2nd sentence semantics) and the position encoding is done on a cell-level, so it restarts from 0 for every cell in body C and header H :

$$\begin{aligned} pos(T) &= (pos(h_i), \dots, pos(c_{i,j})) \\ pos(cell) &= (0, \dots, t) \end{aligned}$$

Similarly as in TURL, we use a table attention mask (visibility matrix) $\alpha_{i,j}$, but on token-level instead of cell-level. This mask allows every token to attend exclusively to tokens which are either in the same row or in the same column. $\alpha_{i,j}$ is a symmetric binary matrix defined as:

$$\alpha_{i,j} = \begin{cases} 1 & \text{if } col(i) = col(j) \vee row(i) = row(j), \\ 0 & \text{otherwise,} \end{cases}$$

where row (col) are functions that map linearized token indices back to row (column) indices in the table.

The output of the token encoder layer is a sequence of token representations:

$$\mathbf{w}_{h_1,1}, \dots, \mathbf{w}_{h_m,t}, \mathbf{w}_{c_{1,1},1}, \dots, \mathbf{w}_{c_{n,m},t} = \text{ENC}(T),$$

which is then fed into a classification layer with a Softmax activation to assign a score for each token to a class $y \in \mathcal{Y}$.

¹huggingface.co token classification models that take a custom 2D attention_mask

4.2 Data Augmentation

As mentioned above, existing DA techniques for NER, such as those presented in (Dai and Adel, 2020), are not a good fit for tabular data, since they produce augmented tables with inconsistent context. For example, the common label-wise token replacement (LWTR) may replace the *QUANT* token *nominal* in Figure 1 with *height* or the *UoM* *bar* with *Celsius*. This clearly introduces inconsistencies in the context, since *height pressure* has no physical meaning and neither *height* nor *pressure* are measured in *Celsius*. A visualization of such an inconsistent table can be seen in the Appendix in Figure 5.

To overcome this problem external domain-specific knowledge is needed. For many industrial domains there exist resources (standardized vocabularies, data models) that can be incorporated for DA. We propose a novel DA approach which leverages existing industrial semantic data models to augment and to generate tables with consistent context. In particular, we use the Reference Data Library (RDL) of POSC Caesar (ISO-15926)². The RDL is a rich source of a domain-specific vocabulary and relations in the process industry. For example, it defines taxonomies that represent specific types of equipment (*EQ*), but also physical quantities (*QUANT*) that plant equipment typically possess. Figure 3 shows a small excerpt of the RDL as knowledge graph. We leverage this data in the process of augmenting existing tables with consistent equipment, quantity and unit of measure (*UoM*) context as follows.

First, we extract surface names (*sfn*) of all entities of type *ENT* into a respective set $RDL_{ENT} = \{sfn-ent_1, sfn-ent_2, \dots\}$, where $ENT \in \{EQ, QUANT, UoM\}$. Additionally, we extract a dictionary $RDL_{EQ} : RDL_{EQ} \rightarrow RDL_{QUANT}$ that holds a set of applicable quantities for every equipment and a second dictionary $RDL_{QUANT} : RDL_{QUANT} \rightarrow RDL_{UoM}$ that stores all applicable units of measure for a certain quantity. The extracted sets for the example graph are also shown in Figure 3.

To ease notation we define a function f_{ner} which returns the set of entity types contained in the set of cells passed as arguments, e.g., $f_{ner}(T_{[:,2]}) = \{EQ\}$ means that the second column of table T contains entities of type *EQ*.

²<http://data.posccaesar.org/rdl/>

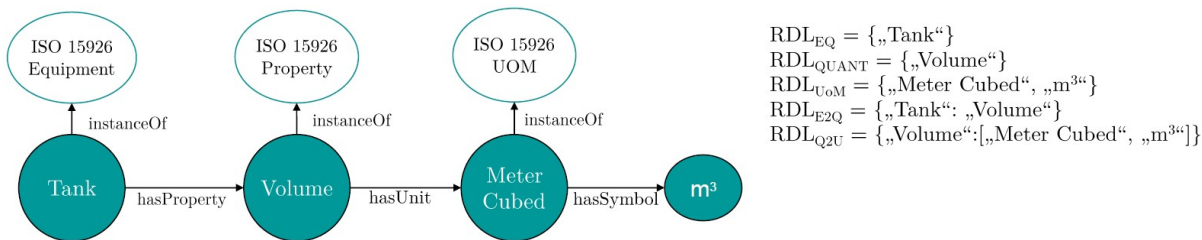


Figure 3: Example graph from POSC Caesar and resulting sets/dictionaries for RDLTab.

Augmentation procedure Given a table T we generate an augmented sample T_{aug} as follows:

1. Sample k columns that contain no NER annotations as starting point for augmentation, $T_{aug} \leftarrow sample(\bigcup_j T_{[:,j]}, k)$, where $f_{ner}(T_{[:,j]}) = \emptyset$.
2. For every row i in T_{aug} : An EQ entity surface name $sfn-eq_i$ is sampled uniformly at random from RDL_{EQ} . The cells in column $k+1$ hold the sampled names: $c_{i,k+1} \leftarrow sfn-eq_i$.
3. Sample a column header h_{eq} from all training table columns that contain at least one EQ annotation: $h_{k+1} \leftarrow h_{eq}$.
4. For every sampled equipment $sfn-eq_i$: a $QUANT$ entity surface name $sfn-quant_i$ is sampled uniformly at random from $RDL_{E2Q}(eq_i)$. Each $sfn-quant_i$ is a new column header in $H_{aug} \leftarrow H_{aug} \cup \{sfn-quant_i\}$. Fill the resp. cells $c_{i,k+i+1}$ with a random numeric value and optionally a randomly sampled UoM surface name from $RDL_{Q2U}(quant_i)$.
5. Finally, generate a last column, where for every sampled equipment $sfn-eq_i$ an artificial TAG entity surface name $sfn-tag_i$ is generated. This column’s header is then sampled from all training tables headers that contain at least one TAG annotation.

Artificial tags are generated by forming an acronym from the EQ entity name and adding groups of random alphanumeric strings, optionally divided by the dash ‘-’ character (which is similar to tagging standards).

5 Experiments

In this section we empirically study the performance of TABNER compared against several baselines as well as the benefits of our domain-specific table DA technique.

tables	μ_{ner}	TAG	EQ	QUANT	UoM
79	18	295	359	427	359

Table 2: Dataset used for experiments.

5.1 Dataset

We extract 79 tables from a pool of real-world spreadsheets describing industrial plant equipment. To get expert labels, we sub-sampled each table to have a maximum of 5 rows. The labels were collected on a cell-by-cell basis using the tool Prodigy³. The statistics of the dataset are shown in Table 2; the mean number of NER-tags per table is 18, the other columns show the absolute number of NER-tags for each entity type. All experiments are carried out in a 5-fold cross validation where we use 10% of each fold’s training data as validation set.

5.2 Baselines

We compare the performance of TABNER to multiple baselines. First, we design a rule-based NER (RULENER) based on spaCy’s EntityRuler⁴ using the same domain-specific vocabularies from RDL as described in section 4.2 for matching. For detecting entities of type TAG we employ a heuristic: find the column with most unique body values which does not contain any known vocabulary terms. Then we mark all alphanumeric tokens as TAG . The second baseline is a BiLSTM-CRF model that uses word embeddings (pre-trained GloVe-6B-100d) as well as character embeddings (Ma and Hovy, 2016). Here, we simply feed each table in linearized form as input. Lastly, we fine-tune a vanilla sequential BERT, again with linearized input tables, without any table structure encoding to study if the tabular structure inductive bias is justified.

³<https://prodi.gy>

⁴<https://spacy.io/api/entityruler>

DA techniques We refer to the DA method explained in section 4.2 as RDLTab and compare its performance against LWTR. For both DA techniques, we experimented with $n = 1, 2$ number of augmented tables per original table in each epoch. In the case of LWTR, we generate n new tables by randomly replacing $m/2$ tokens, where m is the total number of NER labels available for the table. When applying RDLTab, we generate n new tables for every table in the training set. The best performance was achieved with $n = 1$ sample of augmented tables. Therefore, the presented results are with $n = 1$ for both DA techniques and the comparison with the performance when $n = 2$ samples is discussed in the Appendix.

6 Results and Analysis

Convergence First, we analyze the progress of the training loss to study the convergence of the different NER models, see Figure 4. The loss of vanilla BERT is quite flat from the beginning and after a few epochs gets stuck at a bad local optimum - hits early stopping based on validation. We argue that the global attention and position encoding across the full table are blurring the NER training signal for BERT and since we could not find a setting to make it converge properly, we excluded it from further experiments. A more detailed analysis can be found in the Appendix.

In contrast to BERT, the training loss for TABNER is converging quickly. Using only the training data, without augmentation, has the least steepest decline, which is due to observing less labels per epoch. LWTR shows a very steep decline in the beginning which, however, flattens out sooner than RDLTab. Our hypothesis here is that LWTR adds helpful variance in the labels at the beginning, but has less variance to add in the long run, since it can only sample from known training tables. RDLTab on the other hand produces a more novel table context over time as the RDL has richer external vocabulary.

Table structure vs. sequential inductive bias

We present the final cross-validation F1 scores in Table 3. It can be seen that TABNER outperforms the baselines in all DA settings, proving the benefits of being biased towards tabular structure. Surprisingly, BiLSTM-CRF does not suffer from the linearized global table context as much as BERT does and still shows competitive performance. One reason might be that the sequential attention in the

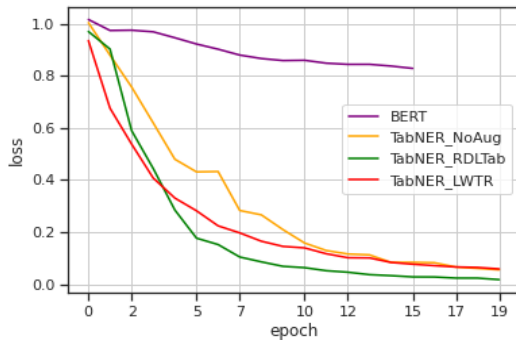


Figure 4: Convergence of the training loss.

Model	NoAug	LWTR	RDLTab
RULENER	0.08	-	-
BiLSTM-CRF	0.53	0.46	0.55
TABNER	0.54	0.52	0.58

Table 3: F1 scores with different DA techniques.

BiLSTM is trained from scratch and can therefore learn to only focus on very narrow context. While BERT is already pre-trained to take long-range context into account.

Data Augmentation The RDLTab DA boosts performance for both TABNER and BiLSTM-CRF. This shows the added value of rich external vocabulary for industrial low-resource problems. Interestingly, LWTR harms performance in both cases. We attribute this to the problem of producing phrases that are non-meaningful physically and inconsistent in a tabular context.

7 Conclusion

In this paper, we demonstrate the applicability of TaLMs to a novel NER problem in industrial spreadsheets. Our experiments show that the tabular inductive bias of TaLMs is not only beneficial for this problem, but may even be a necessary condition when relying on pre-trained transformer-based models. In addition to that we present a DA technique leveraging publicly-available industrial standard information models to produce augmented tables with physically sound and consistent context. Compared to an off-the-shelf DA, this technique shows improved NER performance.

Future work includes understanding how much tabular context is needed to make training large-scale model more efficient. Another fruitful area is active learning for tasks using TaLMs to reduce the time for collecting expert labels.

References

- Vincenzo Cutrona, Federico Bianchi, Ernesto Jiménez-Ruiz, and Matteo Palmonari. 2020. [Tough Tables: Carefully Evaluating Entity Linking for Tabular Data](#).
- Xiang Dai and Heike Adel. 2020. [An analysis of simple data augmentation for named entity recognition](#). In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3861–3867. International Committee on Computational Linguistics.
- Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. [Turl: Table understanding through representation learning](#). *Proc. VLDB Endow.*, 14(3):307–319.
- Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. [Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks](#). In *IJCAI'2022 SURVEY TRACK*.
- Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard H. Hovy. 2021. [A survey of data augmentation approaches for NLP](#). In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 968–988. Association for Computational Linguistics.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisen-schlos. 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.
- Yusra Ibrahim, Mirek Riedewald, and Gerhard Weikum. 2016. [Making sense of entities and quantities in web tables](#). In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM '16*, page 1703–1712.
- Aneta Koleva, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. 2021. [Generating table vector representations](#). In *CEUR Workshop Proceedings - Deep Learning for Knowledge Graphs (DLKG)*.
- Shayne Longpre, Yu Wang, and Chris DuBois. 2020. [How effective is task-agnostic data augmentation for pretrained transformers?](#) In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4401–4411. Association for Computational Linguistics.
- 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, Berlin, Germany. Association for Computational Linguistics.
- 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*. The Association for Computer Linguistics.
- Patrice Y. Simard, Yann LeCun, John S. Denker, and Bernard Victorri. 1996. [Transformation invariance in pattern recognition-tangent distance and tangent propagation](#). In Genevieve B. Orr and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 1524 of *Lecture Notes in Computer Science*, pages 239–27. 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](#). 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.
- Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. [TUTA: tree-based transformers for generally structured table pre-training](#). In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1780–1790. ACM.
- Jason W. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387. Association for Computational Linguistics.
- Sen Wu, Luke Hsiao, Xiao Cheng, Braden Hancock, Theodoros Rekatsinas, Philip Levis, and Christopher Ré. 2018. [Fonduer: Knowledge base construction from richly formatted data](#). In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD '18*, page 1301–1316, New York, NY, USA. Association for Computing Machinery.
- Usama Yaseen and Stefan Langer. 2021. [Data augmentation for low-resource named entity recognition using backtranslation](#). *CoRR*, abs/2108.11703.
- 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.

Sangdoon Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. 2019. [Cutmix: Regularization strategy to train strong classifiers with localizable features](#). In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6022–6031. IEEE.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. [Character-level convolutional networks for text classification](#). In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 649–657.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2020. [Grounded adaptation for zero-shot executable semantic parsing](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6869–6882. Association for Computational Linguistics.

A Appendix

DA example Figure 5 shows how tabular context becomes inconsistent when applying LWTR to the table in Figure 1. The red tokens have been replaced with sampled tokens from the training set. It can be seen that the *QUANT* entity *height pressure* is now physically meaningless and neither *height* nor *pressure* are measured in *Celsius*.

Probing table context To demonstrate the sensitivity of TABNER towards table context, we construct two synthetic tables with slightly modified cell content. The table at the top in Figure 6a has a column with header *power (QUANT)* with body cells having random (inconsistent) *UoM* entities *8 celsius* and *90 l*. The bottom table’s column with header *capacity* has consistent *UoM* context. We are interested in how these two different contexts affect the classification of token ‘*l*’, which is hard to classify without context. In a column like *capacity* it likely refers to the *UoM* entity ‘*liter*’. However, in most other contexts ‘*l*’ is not part of any entity. Looking at the respective logits in Figure 6b, we can see that TABNER is sensitive to these context changes. The highest scoring class for the random context is *O*, while in the consistent case it is the class *UoM*. This is a beneficial property, since it prevents false positives for highly ambiguous tokens such as ‘*l*’, which only in very specific contexts are likely to be entities.

Experiment Details For fair comparison, both TABNER and BERT are based on the pre-trained

Model	TAG	EQ	QUANT	UoM
RULENER	0.1	0.09	0.04	0.1
BiLSTM-CRF	0.55	0.39	0.54	0.67
TABNER	0.60	0.43	0.47	0.77

Table 4: Class-wise F1 scores.

‘bert-base-uncased’ and we select the best hyperparameters from these ranges: learning rate $\{5e^{-5}, 1e^{-5}, 5e^{-4}\}$, batch size $\{2, 4, 8\}$. The learning follows a linearly decreasing schedule with a maximum of 20 epochs. For the BiLSTM-CRF we use the NER hyperparameters from (Ma and Hovy, 2016).

BERT Analysis In our experiments, we observe that BERT almost exclusively fits to the *O* token labels in the training set and does not pick up on the other NER signals. Since it is an imbalanced problem, our hypothesis is that the global attention and position encoding across the full table blurs tokens with less frequent NER signals and BERT cannot properly fit them. More sophisticated weighted loss functions could be tried to remedy this problem. In Figure 7 the progress of the validation set F1 score is shown. Even though the training loss is still slightly decreasing, the validation NER performance seems to have already peaked. In all hyperparameter settings (even with much lower learning rate $1e^{-7}$) we could not achieve a test F1 score higher than 0.03.

Class-wise F1 scores As more fine-grained analysis, we present the class-wise F1 scores for each model in Table 4. We can see that the TABNER is better in extracting entities of types *TAG*, *EQ* and *UoM*, while the BiLSTM model is better at classifying entities of type *QUANT*.

Data Augmentation Samples We experiment with $n = 1, 2$ samples to evaluate if increasing the training set by more than 100% will bring benefit to the TabNER model. Figure 7 shows the validation set F1 score for the TabNER model with the two DA techniques, LWTR and RDLTab, and the different $n = 1, 2$ samples. Consistently, for both techniques, when $n = 2$ the model converges much faster, after only 5 epochs, however the performance of the model is worse compared to when we use $n = 1$.

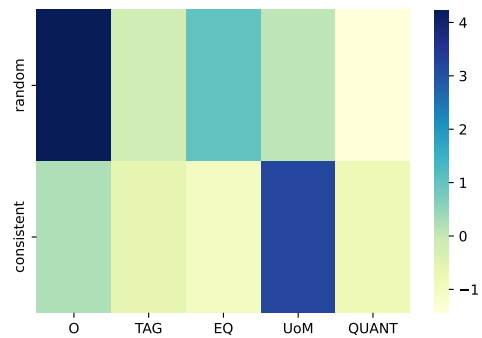
No.	Eq.name	Risk rating	min/max height pressure	Pow rating (kW)	Ow. Code
P-47-01	Area 47 - Cent. Valve for refinement	High	20/30 Celsius Pump 10/20 bar Pipe	20	5MW
T-41-02/A, T-41-02/B	Wtr Tank (qty: 2)	Low	Inner: 10/20 Outer: 5/10	- Capacity : 500 l	2AV

Figure 5: LWTR introduces inconsistent tabular context. Red tokens have been replaced in the original in Figure 1.

eq	power	Other
External valve	8 celsius	Open
Wtr pump	90 l	Closed

eq	capacity	Other
External valve	8 m3	Open
Wtr pump	90 l	Closed

(a) Two synthetic tables with small modifications. Top has random context, bottom has consistent context.



(b) Unnormalized logits for token 'l' in top and bottom table in 6a.

Figure 6: TABNER token logits with synthetic consistent and randomized table context.

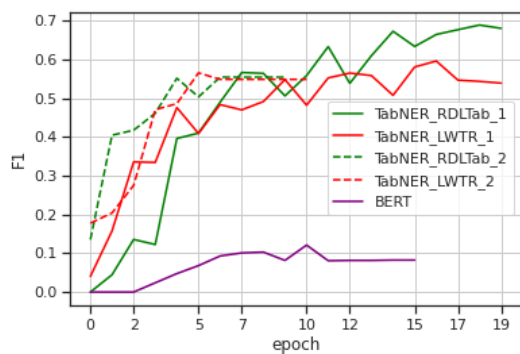


Figure 7: F1 score on validation set during training.

Reinforced Question Rewriting for Conversational Question Answering

Zhiyu Chen, Jie Zhao, Anjie Fang, Besnik Fetahu, Oleg Rokhlenko, Shervin Malmasi
Amazon.com, Inc., Seattle, WA, USA

{zhiyu, zhaojie, njfn, besnikf, olegro, malmasi}@amazon.com

Abstract

Conversational Question Answering (CQA) aims to answer questions contained within dialogues, which are not easily interpretable without context. Developing a model to rewrite conversational questions into self-contained ones is an emerging solution in industry settings as it allows using existing single-turn QA systems to avoid training a CQA model from scratch. Previous work trains rewriting models using human rewrites as supervision. However, such objectives are disconnected with QA models and therefore more human-like rewrites do not guarantee better QA performance.

In this paper we propose using QA feedback to supervise the rewriting model with reinforcement learning. Experiments show that our approach can effectively improve QA performance over baselines for both extractive and retrieval QA. Furthermore, human evaluation shows that our method can generate more accurate and detailed rewrites when compared to human annotations.

1 Introduction

Interacting through conversations is a natural information-seeking procedure for humans, therefore it is important for AI assistants like Apple Siri and Amazon Alexa to enable and improve such experiences. In recent years Conversational Question Answering (CQA) has gained more attention, where a user can ask a series of related questions and ideally obtain answers that leverage the conversational context. Different from widely-studied question answering (QA) tasks that happen in single-turn (Rajpurkar et al., 2016, 2018; Tay et al., 2018; Tang et al., 2019), the interpretation of conversational questions in CQA depends on questions and answers from previous turns.

Previous approaches to CQA usually train new models from scratch, which can achieve promising results but also are expensive in terms of obtaining domain-specific training data. In industry settings,

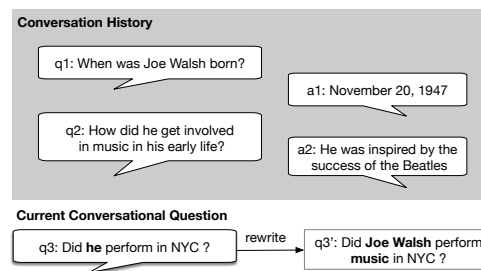


Figure 1: A conversational question rewriting example.

there are many single-turn QA models deployed. Training new CQA models with additional annotations to replace each existing single-turn QA model is expensive, and generally not feasible. Moreover, discarding existing single-turn models and datasets is impractical, and studying how to reuse these existing resources to tackle CQA merits attention.

Existing approaches to this task, called Conversational Question Rewriting (CQR), often train sequence-to-sequence models supervised by human rewrites to generate self-contained questions (Ren et al., 2018; Vakulenko et al., 2021). Such methods have several limitations. First, the CQR training objective is disconnected from CQA performance. The annotation process of existing rewriting datasets has no knowledge of the QA systems, and more human-like rewrites do not guarantee better CQA performance. Second, the rewriting model does not take into account the feedback from downstream QA systems. In industry settings, multiple single-turn QA systems trained with different datasets serve in the backend. It is impractical to replace them with new CQA models, and we argue that their output can still be used as signals to help train rewriting models.

To overcome these limitations, we propose an effective CQR approach upon the recent success of Reinforcement Learning (RL) techniques for text generation (Rennie et al., 2017). RL enables flexible ways to incorporate training objectives in the

form of reward functions. We systematically analyze different rewards and their effectiveness in terms of final QA performance, as well as the quality of the question rewrites (i.e. the question still has to be understandable and interpretable by humans). To optimize QA performance, we propose various QA rewards to measure the likelihood of a question yielding a better answer. In comparison with the QA rewards, we also propose to use the same RL approach with question rewriting (QR) rewards reflecting the similarity between a model-generated question and the human’s ground-truth.

We summarize our contributions as follows:

- To the best of our knowledge, we are the first to study how to incorporate QA signals to improve CQR using RL.
- We systematically propose and compare using different training signals as rewards.
- We conduct experiments on two CQA tasks to show our approach is effective.
- A user study shows that our method can generate more accurate and detailed rewrites when compared to human annotations.

2 Related Work

Conversational Question Answering. Recently, conversational QA has been studied which presents new challenges for QA models such as being able to resolve conversational dependencies so that a conversational question can be interpreted by QA models with conversational context. QuAC (Elgohary et al., 2019) and CoQA (Reddy et al., 2019) are two datasets for extractive CQA where answers are extracted from passages. CAsT-19 (Dalton et al., 2020) is a benchmark for retrieval CQA and the target is to return relevant passages given a question. QReCC (Anantha et al., 2021) combines retrieval and extractive CQA where the answers are extracted from passages returned by a retrieval component. Kim et al. (2021) propose to train the CQA model and rewriter simultaneously, which is impractical for industry setting. A directly related work to ours is Vakulenko et al. (2021) which proposes to rewrite questions for CQA. However, they do not consider taking the QA feedback into the CQR training which is studied in our work.

RL for Nature Language Generation. Reinforcement learning methods have been widely applied for various language generation tasks. Li et al. (2016) propose to apply deep reinforcement learning in dialogue generation to model future rewards

related to conversational properties, such as informativeness, coherence and ease of answering. Ranzato et al. (2016) propose Mixed Incremental Cross-Entropy Reinforce (MIXER) for sequence prediction to directly optimize the metrics used at test time, such as BLEU or ROUGE. They show MIXER outperforms several strong baselines for greedy generation on text summarization, image caption and machine translation. Nogueira and Cho (2017) train a query rewriter based on the rewards relying on the ground-truth ranking list for information retrieval. Buck et al. (2018) use RL for single-turn question rewriting by maximizing the answers’ quality which requires ground-truth. Similar to our F1 reward, Wu et al. (2021) design rewards from ground-truth answers to train a conversational query rewriter. Instead, we propose alternative rewards indicating the confidence of answers from a model itself which do not require human annotations.

3 Problem Definition

In CQA, each conversation contains a sequence of (question, answer) pairs $D = \{q_1, a_1, \dots, q_n, a_n\}$, where a_i is the answer for question q_i . A conversational question q_i can be ambiguous and its interpretation depends on the conversational context $c_i = \{q_1, a_1, \dots, q_{i-1}, a_{i-1}\}$. The goal of CQR for QA is to learn a model \mathcal{R}_θ , parameterized by θ , that can translate q_i associated with c_i into q'_i , so that the semantic meaning of q'_i is equivalent to q_i .

$$q'_i = \mathcal{R}_\theta(q_i, c_i) \quad (1)$$

A pretrained single-turn QA model is expected to answer q'_i better than q_i . Note that the QA model can be trained from a single-turn dataset different from D and is fixed when training the rewriter. The motivation is to explore whether the already deployed single-turn QA models can be exploited to train a rewriter and reused without further training by accepting the rewritten questions.

4 Approach

4.1 Model Overview

We show our CQR approach with a modularized design in Figure 2. There are two major components: a CQR model \mathcal{R}_θ as introduced in Section 3 and a reward function \mathcal{F} that evaluates rewrite q'_i generated by \mathcal{R}_θ by producing a reward score. Then the CQR training can be formulated as a reinforcement training problem, where the objective

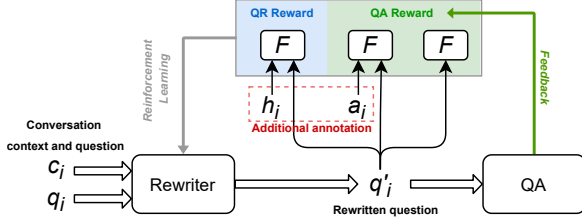


Figure 2: Overview of our CQR approach. h_i is human rewriting of q_i and a_i is the ground-truth answer of q_i .

is to maximize an expected reward or equivalently minimize the following loss function:

$$\mathcal{L}_{rl}(\theta) = -\mathbb{E}_{q'_i \sim \mathcal{R}_\theta(q_i, c_i), q_i \sim \mathcal{T}}(\mathcal{F}(q'_i)), \quad (2)$$

where q_i comes from data distribution \mathcal{T} . During training, we push \mathcal{R}_θ to generate q'_i that achieves a higher reward by minimizing Equation 2. Hereinafter, we omit θ from \mathcal{R}_θ for simplicity.

We define two types of rewards: QR rewards evaluate how similar a question rewrite is to the ground truth one produced by human annotators; QA rewards evaluate how well a QA model can answer a question rewrite. We summarize the characteristics of different rewards in Table 1. By maximizing one of the QR or QA rewards, we can explicitly optimize the model to achieve the QR or QA target. Next, we describe the two types of rewards.

Reward	ROUGE	F1	Confidence	BM25
Reward Type	QR	QA	QA	QA
CQA Type	-	Extractive	Extractive	Retrieval
Need Annotated Rewrites	Y	N	N	N
Need Annotated Answers	N	Y	N	N

Table 1: Characteristics of different rewards.

4.2 QR Rewards

The rationale of maximizing QR rewards is similar to the aims of prior work: a good question rewrite should be similar to a human rewrite. We use the ROUGE-L score (Lin, 2004) between the question rewrite q'_i and the ground-truth h_i as the QR reward:

$$\mathcal{F}(q'_i, h_i) = \text{ROUGE}_L(q'_i, h_i) \quad (3)$$

This reward has been widely used by RL methods for language generation tasks. Note that Eq. 3 does not depend on the QA model and prior work can be considered as maximizing QR rewards.

4.3 QA Rewards

We define QA rewards that reflect how well the question rewrites can help a QA model obtain better answers. Since QA rewards are task/model-dependent, we introduce QA rewards for the following two sub-types.

4.3.1 Extractive CQA

Extractive CQA is a machine reading comprehension (MRC) task and an extractive QA model \mathcal{M} extracts the most likely span answer given a question q and an evidence document p :

$$a_s = \arg \max_{a_s} P_{\mathcal{M}}(a_s | q, p) \quad (4)$$

We assume that \mathcal{M} is trained on regular single-turn QA data, and expects the input question q to be self-contained. Therefore, CQA questions should be rewritten by \mathcal{R} before being sent to \mathcal{M} . Next, we introduce supervised and unsupervised QA rewards.

Supervised QA rewards. A straightforward way to measure the quality of a question rewrite q'_i in terms of QA is to calculate the similarity between the predicted answer by \mathcal{M} with q'_i as input and the ground-truth answer a_i . We denote a'_s as the extracted answer span by \mathcal{M} using the rewritten question q'_i as input. We measure the overlap between a'_s and a_i by F1 score:

$$\mathcal{F}(q'_i, a_i) = F1(\arg \max_{a'_s} P_{\mathcal{M}}(a'_s | q'_i, p), a_i) \quad (5)$$

Intuitively, the rewrite q'_i is better if a'_s is closer to the ground-truth answer. Compared with Equation 3, Equation 5 depends on the ground-truth answers instead of human rewrites.

Unsupervised QA rewards. For a predicted span a'_s , \mathcal{M} assigns a probability $r_c = P_{\mathcal{M}}(a'_s | q'_i, p)$ that reflects the model’s confidence about the answer. We assume that a higher confidence score of an answer indicates that the QA model has a better question understanding. Therefore, we directly use the probability of the most likely answer as the confidence reward for a question rewrite:

$$\mathcal{F}(q'_i) = \max P_{\mathcal{M}}(a'_s | q'_i, p) \quad (6)$$

F1 rewards can be considered as judgment scores on predicted answers by humans since the ground-truth answers are used, while confidence rewards represent the model’s self-judgments.

4.3.2 Retrieval CQA

We also evaluate our method’s generalization on a different **retrieval CQA** task, where the goal is to return a list of documents in descending order of relevance scores produced by a retrieval CQA model:

$$rel = \mathcal{M}(q, p) \quad (7)$$

where p is a document. A retrieval CQA model usually consists of two stages. In the first stage, a lightweight ranking algorithm such as BM25 is used to retrieve top-k candidate documents. In the second stage, a more complex model such as BERT (Devlin et al., 2019) is used to rerank candidate documents. Here, we use the BM25 score between a question and a document, which is a type of QA reward that does not use annotated answers:

$$\mathcal{F}(q'_i) = BM25(q'_i, p) \quad (8)$$

We expect the rewrite q'_i can retrieve documents with higher BM25 scores in the first stage than q_i so that the performance in the re-ranking stage can also be improved.

4.4 Training

There are two steps in our training framework. The first step, the pre-training step, which has the same supervised target as prior work. The objective is to minimize the cross-entropy loss between the model’s prediction q' and human ground-truth rewrites h :

$$\mathcal{L}_{sup} = -y_h \log y_{q'} , \quad (9)$$

where y_h is the one-hot vector of h and $y_{q'}$ is the distribution over tokens in q' predicted by the model. Supervised pre-training ensures the model has the basic ability to rewrite the original question given the conversational context.

The second step continues training \mathcal{R} with RL to maximize different rewards. In this work, we use Self-Critical Sequence Training (SCST) (Rennie et al., 2017). Given a question q , we generate two question rewrites q^s and q' . q^s is generated by sampling the word distribution from \mathcal{R} at each step, and q' is generated by \mathcal{R} using greedy decoding. Then we minimize the following loss function:

$$\mathcal{L}_{rl} = (r' - r^s) \sum_{t=1}^N \log P_{\mathcal{R}}(w_t^s | w_{1:t-1}^s, q, c) \quad (10)$$

Here, $P_{\mathcal{R}}(\cdot)$, which is defined by \mathcal{R} , is the probability of generating the t -th word conditioning on previously generated tokens of q_s , the original question q and conversational history c . Intuitively, minimizing \mathcal{L}_{rl} increases the likelihood of q^s if it obtains a higher reward than q' (i.e. $r^s > r'$), and thus maximizes the expected total reward. Given a reward function, we can obtain $r' = \mathcal{F}(q')$ (\mathcal{F} can be one of Equation 3,5,6,8) and $r^s = \mathcal{F}(q^s)$.

We only choose one of the reward functions to obtain the reward for a question. We leave the combination of different rewards as future work. Additional training procedure details are described in Appendix A.

5 Data and Experimental Setup

5.1 Datasets

Similar to Vakulenko et al. (2021), we experiment with CANARD (Elgohary et al., 2019) for extractive CQA and CAsT-19 (Dalton et al., 2020) for retrieval CQA. As CAsT-19 is small compared to CANARD, prior work (Vakulenko et al., 2021) uses the same model trained on CANARD to evaluate the rewriting performance on the test set of CAsT-19. Similarly, we start with the model on CANARD, and continue RL training with the BM25 reward on the training set without using any human annotations provided by CAsT-19.

5.2 Evaluation Metrics

We use BLEU-1, BLEU-4, ROUGE-1 and ROUGE-L for automatic evaluation. We also evaluate the performance of rewrites on downstream QA tasks. For CANARD, we use F1 and Exact Match (EM). For CAsT-19, we report MAP, MRR and NDCG@3 as in Vakulenko et al. (2021).

5.3 Baselines

We consider the following baselines:

Origin uses the original conversational question as input of QA.

BART_{CQR} We fine-tune BART (Lewis et al., 2020) as a supervised baseline which has the same training procedure as the pre-training step of our method.

Co-reference (Vakulenko et al., 2021) is a rule-based method. We replace anaphoric expressions in original questions with their antecedents from the previous conversation turns. A public neural co-reference model (Lee et al., 2018) is used.

QR Method	QA Metrics		QR Metrics			
	EM	F1	B-1	B-4	R-1	R-L
Human	42.41	54.53	-	-	-	-
Original	38.41	48.95	61.06	30.98	69.91	69.71
Co-reference	38.17	48.99	54.95	30.84	74.11	73.40
BART _{CQR}	41.26	53.60	64.20	39.33	76.70	74.00
RL-QR	41.33	53.74	64.25	39.52	76.70	74.01
RL-F1	41.91	54.27 [†]	62.32 [†]	37.79 [†]	74.93 [†]	72.09 [†]
RL-C	41.91	54.61 [†]	57.47 [†]	34.18 [†]	71.12 [†]	68.21 [†]

Table 2: Overall QR and QA performance (%) on CANARD. **Bold** indicates the best results except “Human”. We denote BLEU-n as B-n and ROUGE-n as R-n. [†] denotes statistically significant difference from BART_{CQR} ($p < 0.05$ with t-test).

QR Method	QA Metrics		QR Metrics			
	EM	F1	B-1	B-4	R-1	R-L
BART _{CQR} (50%)	41.37	53.52	63.83	38.88	76.57	73.79
RL-C (50%)	42.09	54.76 [†]	62.13	37.52	75.03	72.10
RL-C (50%+100%)	42.05	54.84 [†]	57.86 [†]	34.44 [†]	71.67 [†]	68.54 [†]

Table 3: QR and QA performance (%) of BART_{CQR} and RL-C when using 50% of ground-truth rewriting. [†] denotes statistically significant difference from BART_{CQR} (50%) ($p < 0.05$ with t-test).

Human uses the human rewrites and can be considered as an upper bound. However, we later show that the human baseline is the upper bound for QR target but not QA target.

5.4 Implementation Details

For all the QA models, we simulate the scenario where they are trained on single-turn QA data and cannot be updated when interacting with the rewriting component. The goal is to improve single-turn QA models for CQA, which means the input for QA models does not include any previous context.

Single-turn Extractive QA Model. To simulate a single-turn extractive QA model, we fine-tune ALBERT-XXLarge-v2 (Lan et al., 2020) on the CANARD training set.

Single-turn Retrieval QA Model. Same as in Vakulenko et al. (2021), we use Anserini’s implementation of BM25 (Robertson et al., 2009) for the first-stage retrieval to obtain the top 1000 passages. In the second stage, we use BERT-large for passage re-ranking. Both components are fine-tuned on the MS MARCO dataset so that the two-stage pipeline resembles a single-turn retrieval QA model.

Rewriting Models. Our RL-based methods and the supervised BART baseline (BART_{CQR}) use

BART-base model (Lewis et al., 2020).¹ We use the official CANARD validation set for early stopping. **RL-QR** denotes the model when QR rewards are used. **RL-F1**, **RL-C** and **RL-BM25** denote models where the F1, confidence and BM25 rewards are used, respectively.

6 Results

Here, we study the following research questions:

RQ1: Can our proposed QR and QA rewards improve the overall CQA performance? In particular, how effective are unsupervised rewards?

RQ2: Does achieving the best QR target mean achieving the best QA target?

RQ3: What is the quality, as judged by humans, of the reward-guided question rewrites?

6.1 Evaluation on Extractive CQA

We list the results on CANARD in Table 2. EM and F1 are QA metrics while others are QR metrics. We observe several trends.

First, RL-based methods achieve the best results on both QA or QR metrics over other non-human baselines. Compared with BART_{CQR}, our proposed RL methods can further improve the per-

¹The max sequence length is set to 284, with batch size 24. An Adam weight decay optimizer with an initial learning rate of 1e-5 is used to train those models for 10 epochs.

QR Method	QA Metrics			QR Metrics			
	MAP	MRR	NDCG@3	B-1	B-4	R-1	R-L
Origin	17.85	46.44	27.86	71.63	51.54	82.65	81.24
Human	39.23	87.06	58.19	-	-	-	-
BART _{CQR}	28.02	61.49	44.04	75.12	55.54	84.82	83.84
Co-reference	26.82	59.74	43.05	71.19	51.79	88.06	87.69
RL-BM25	28.41	63.20	45.54	71.92	52.01	82.92	81.59

Table 4: QR and retrieval performance (%) on CAsT-19.

formance on QA target and QR target. Specifically, RL-C outperforms BART_{CQR} by 1.88% and 1.58% in terms of F1 and EM, respectively. RL-QR achieves marginally better scores on BLEU-1, BLEU-4 and ROUGE-L than BART_{CQR}. RL-F1 achieves better F1 and EM scores than RL-QR and BART_{CQR} but does not outperform RL-C. We notice that the F1 reward is less sensitive to question rewrites than the confidence reward. A small change in a question can lead to the same answer and F1 score. However, the confidence score can be different. In this aspect, RL-C seems to differentiate the fluctuations on rewrites better than RL-F1. In answer to **RQ1**, the confidence reward is the most effective for CQA performance. As an unsupervised reward which does not require either human rewrites or gold answers to a question, the confidence reward is even more effective than the F1 reward. However, we do not claim or target state-of-the-art performance in our work. The goal is to verify whether our RL framework for CQR with different rewards can further improve the performance of a single-turn QA system for CQA.

Second, using QR rewards (RL-QR) leads to limited performance improvement under both QA and QR metrics compared with BART_{CQR}. Maximizing the ROUGE rewards (Eq. 3) and minimizing the cross-entropy loss (Eq. 9) share the similar intuition that a good reformulation from the model should be similar to human reformulated questions. The two objectives are very close and therefore lead to similar results. It is important to note that the best scores of QR metrics and QA metrics are not achieved by the same method. Moreover, using QA rewards even lead to a large decrease in QR metrics. Therefore, in response to **RQ2**, achieving the best QR target does not mean achieving the best QA target, and vice versa.

Third, **RL-C achieves higher F1 scores than the human baseline**. Previous work (e.g. Vaku-

lenko et al., 2021) treats human annotations as an upper bound. However, we argue that more human-like rewrites do not guarantee better QA performance. The results verify our hypothesis that QA target does not necessarily align with QR target. In §6.4, we qualitatively analyze if rewrites generated by RL-C are better than the ground-truth.

6.2 Training with Fewer Samples

For a real-world CQA system, we can obtain a large number of user questions with no corresponding ground-truth rewrites or answers. Since the confidence reward can be obtained easily from the downstream QA models without requiring human annotations, we can use RL-C to continue training the rewriting model. We first train a baseline using 50% of training data from CANARD (denoted as BART_{CQR} (50%)). Then we continue RL training with the confidence reward using either the same 50% data used in pre-training (denoted as RL-C (50%)) or all questions in CANARD training set (denoted as RL-C (50%+100%)). The results are summarized in Table 3. We can see that RL-C (50%+100%) benefits from the large amount of questions during RL training and achieves better F1 and EM scores than RL-C (50%). Interestingly, RL-C (50%+100%) outperforms the human baseline in Table 2 by 0.31% in terms of F1. We also experimented with other ratios of data for supervised pre-training and continually RL training. In the experiments, we had similar observations that continual RL training with confidence rewards can further improve the downstream CQA performance.

6.3 Evaluation on Retrieval CQA

For RL-BM25, we use RL-C trained on CANARD as the pretrained model, then train it to maximize the BM25 reward, which can be readily obtained from the retrieval model. Results on CAsT-19 are shown in Table 4. As with extractive CQA, RL-BM25 achieves lower scores on QR metrics than

	RL-C vs. BART _{CQR} (%)	RL-C vs. Human (%)
(1) RL is better	121 (60.5%)	105 (52.5%)
(2) RL is worse	39 (19.5%)	58 (29.0%)
(3) Both are good	28 (14.0%)	33 (18.5%)
(4) Both are bad	12 (6.0%)	4 (2.0%)
Total	200	200

Table 5: Results of user study comparing two groups of rewrites using four preference options.

Example 1	Original	What happened after he was fired?
	Human	What happened after Aynsley Dunbar was fired?
	BART _{CQR}	What happened after Aynsley Dunbar was fired?
	RL-C	What happened after Aynsley Dunbar was fired by Herbie Herbert in late 1978 ?
Example 2	Original	What position did he play?
	Human	What position did Red Schoendienst play?
	BART _{CQR}	What position did <u>Ernie</u> Schoendienst play?
	RL-C	What position did <u>Don</u> Schoendienst play in the Majors ?

Table 6: Qualitative comparison of question rewrites. More examples are shown in Appendix C.

baselines. However, it improves the NDCG@3 of BART_{CQR} by relatively 3.4%, which shows our framework also generalizes to retrieval CQA. Note that we do not use any supervised signals in CAS-T-19 training set for RL training.

6.4 Human Evaluation

In addition to CQA performance, generating user-friendly rewrites is also important for real-world applications, since the rewrites sometimes will be displayed to users. To answer **RQ3**, we perform a user study to evaluate the quality of model generated rewrites. Specifically, two groups are compared: (1) The first group contains the rewrites generated by RL-C and human rewrites; (2) The second group contains rewrites from RL-C and BART_{CQR}, respectively. For each group, we randomly choose 200 questions from CANARD testing set. For each pair, we collect human’s judgments on which rewrite contains more accurate context and details from conversation history.

The results are shown in Table 5. The study suggests that RL-C significantly performs better than Human and BART_{CQR} (p-value < 0.001, see details in Appendix B.2). Remarkably, annotators prefer the rewrites from RL-C than humans in more than 50% cases. We show two examples in Table 6. In the first example, both RL-C and BART_{CQR} correctly replace the pronoun with the referred person name. However, the rewrite generated by RL-C includes more accurate details which appear in conversation history. In the

second example, both RL-C (same as RL-F1 and RL-QR) and BART_{CQR} fail to generate the correct person’s name. This error might be due to the prior knowledge of BART. To answer **RQ3**, we find that our reward-guided model can generate rewrites preferred by humans. However, all rewriting models can suffer from the coreference resolution problem.

7 Conclusion

We proposed a conversational question rewriting (CQR) approach using reinforcement learning. Such rewriting approaches are an emerging solution in real-world settings where QA systems with many existing answering backends trained on standalone questions must be adapted to work in conversational settings.

After assessing various QA and QR rewards, we showed that optimizing QR rewards is limited in improving CQA performance. In contrast, QA rewards that do not require ground-truth annotations consistently achieve the best CQA performance over baselines. For extractive CQA, using confidence rewards improved F1 by 2% over BART-based baseline on CANARD; and for retrieval CQA, using BM25 rewards improved the NDCG@3 of the baseline by 3.4% on CAS-T-19. A human evaluation also demonstrated that our approach can generate higher-quality rewrites with more accurate and detailed context information.

References

- Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. 2021. [Open-domain question answering goes conversational via question rewriting](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 520–534, Online. Association for Computational Linguistics.
- Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. [Ask the right questions: Active question reformulation with reinforcement learning](#). In *International Conference on Learning Representations*.
- Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2020. Trec cast 2019: The conversational assistance track overview. *ArXiv*, abs/2003.13624.
- 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.
- Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. [Can you unpack that? learning to rewrite questions-in-context](#). 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 5918–5924, Hong Kong, China. Association for Computational Linguistics.
- Gangwo Kim, Hyunjae Kim, Jungsoo Park, and Jaewoo Kang. 2021. [Learn to resolve conversational dependency: A consistency training framework for conversational question answering](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6130–6141, Online. 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*.
- 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: Human Language Technologies, Volume 2 (Short Papers)*, pages 687–692, New Orleans, Louisiana. Association for Computational Linguistics.
- 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.
- Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. [Deep reinforcement learning for dialogue generation](#). In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.
- 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.
- Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583.
- 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.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. [Sequence level training with recurrent neural networks](#). In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.
- Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. 2018. Conversational query understanding using sequence to sequence modeling. In *Proceedings of the 2018 World Wide Web Conference*, pages 1715–1724.
- Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024.

- Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Min Tang, Jiaran Cai, and Hankz Hankui Zhuo. 2019. Multi-matching network for multiple choice reading comprehension. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7088–7095.
- Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 583–591.
- Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question rewriting for conversational question answering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 355–363.
- Zequiu Wu, Yi Luan, Hannah Rashkin, David Reitter, and Gaurav Singh Tomar. 2021. Conqr: Conversational query rewriting for retrieval with reinforcement learning. *arXiv preprint arXiv:2112.08558*.

Appendix

A Algorithm

There are two steps for training the rewriting model.

1. The 1st step pre-training (line 1-6) is to minimize the cross-entropy loss between the model’s prediction q' and human ground-truth rewrite h . This objective is used in most of prior work (e.g., Vakulenko et al. (2021)).
2. The 2nd step (line 8-16) continues training the model with a reinforcement learning method (Self-Critical Sequence Training). In line 10 and 11, we only chose one of the reward functions to obtain the reward for a question. We leave the combination of different rewards as future work.

Algorithm 1: CQR training

```
Input : Initialized rewriter  $\mathcal{R}$ , human question
rewrites  $H$ , conversations  $\mathcal{T}_{pre}$  for
pre-training, conversations  $\mathcal{T}_{rl}$  for RL
training, selected reward function  $\mathcal{F}$ 
Output : Trained rewriter  $\mathcal{R}$ 
/* Step 1: pre-training  $\mathcal{R}$  */
1 for  $D \in \mathcal{T}_{pre}$  do
2   for question  $q$ , context  $c \in D$  and  $h \in H$  do
3     generate  $q' = \mathcal{R}(q, c)$  (greedy decoding);
4     minimize loss in Equation 9;
5   end
6 end
7 /* Step 2: self-critical training */
8 for  $D \in \mathcal{T}_{rl}$  do
9   for question  $q$ , context  $c \in D$  and  $h \in H$  do
10    generate  $q^s$  from  $\mathcal{R}(q, c)$  by sampling;
11    generate  $q'$  from  $\mathcal{R}(q, c)$  by greedy
    decoding;
12    obtain  $r^s = \mathcal{F}(q^s)$ ;
13    obtain  $r' = \mathcal{F}(q')$ ;
14    minimize loss in Equation 10;
15   end
16 end
```

B Human Study Design

For each annotation, an annotator is presented with the evidence document, conversation history, the original question and two rewrites. The annotator is required to select one from four options as listed in Table 5. The source of rewrite is anonymized. For each pair of rewrite, we randomly assign them to two options so that the judgments are not biased by the position of choices. We collect two judgments

per rewrite pair. If there is a tie, we collect additional judgments. The final judgments are based on majority vote.

B.1 Appen Interface

Figure 3 shows the interface for annotators. Figure 4 contains the instruction which is visible for each annotator. In the instruction, we show several annotation examples in Figure 5.

B.2 Significance Tests

Here we describe how we conduct the Wilcoxon signed-rank test on the annotation results. When comparing RL-C with Human, for each sample, if annotators think RL-C is better, RL-C obtains score 1 and Human obtains score -1. Similarly, if annotators think Human is better, then Human obtains score 1 and RL-C obtains score -1. For other cases (i.e. both are good or both are bad), each of them obtains score 0. Then we use the method “scipy.stats.wilcoxon” in scipy library² to do the test. About the study annotator agreement rates, 48% samples have 100% agreement and the overall agreement rate is around 80%.

C Rewriting Examples

In Table 7, we show examples where the rewrites generated by RL-C are preferred by human annotators over the baseline method and ground truth. Compared with ground-truth rewrites, RL-C tends to generate rewrites with more factual details, which can help the user and also downstream QA systems to understand the question without conversation history. To some degree, it explains why the CQA performance is improved with RL-C, while the corresponding scores of QR metrics (i.e., BLEU-1, BLEU-4, ROUGE-1 and ROUGE-L) are very low. It also indicates that the human ground-truth in existing CQR datasets is not perfect and only evaluating CQR model with QR metrics can be biased.

The cases where both RL-C and the baseline generate incorrect rewrites are shown in Table 8. We can see that both methods make mistakes in coreference resolution. However, RL-C still has the tendency to include more conversational context in the rewrites.

²<https://docs.scipy.org>

Original	What kind of things did she write about?
Human	What kind of things did Le Guin write about?
BART _{CQR}	What kind of things did Le Guin write about?
RL-C	What kind of things did Le Guin write about in Sociology ?
Original	What did he study at the university?
Human	What did Chinua Achebe study at the university?
BART _{CQR}	What did Chinua Achebe study at the university?
RL-C	What did Chinua Achebe study at the University of Ibadan ?
Original	What did he do after leaving Arrowverse?
Human	What did John Barrowman do after leaving Arrowverse?
BART _{CQR}	What did John Barrowman do after leaving Arrowverse?
RL-C	What did John Barrowman do after leaving the Arrowverse television franchise ?
Original	What kind of topics did the show cover?
Human	What kind of topics did the Rush Limbaugh Show cover?
BART _{CQR}	What kind of topics did Rush Limbaugh's show cover?
RL-C	What kind of topics did Rush Limbaugh's radio show cover in the 1970s ?
Original	What did he do after he landed?
Human	What did Lindbergh do after he landed?
BART _{CQR}	What did Charles Lindbergh do after he landed at Le Bourget Aerodrome ?
RL-C	What did Charles Lindbergh do after he landed at Le Bourget Aerodrome at 10:22 p.m. on Saturday, May 21, 1927 ?

Table 7: Examples of rewrites where the reformulated questions from RL-C are the best judged by human annotators.

Original	Did others agree with him ?
Human	Did others agree with Gottfried Wilhelm on the idea that the truth of religion and philosophy cannot contradict with each other?
BART _{CQR}	Did others agree with <u>Leibniz's Theodicy</u> ?
RL-C	Did others agree with <u>Leibniz's</u> ideas that the truths of theology (religion) and philosophy cannot contradict each other, since reason and faith are both "gifts of God" in the Theodicy?
Original	What did he rejoin as?
Human	What did Eddie Collins rejoin as?
BART _{CQR}	What did <u>Chris Hedges</u> rejoin as?
RL-C	What did <u>Chris Hedges</u> rejoin as in August?
Original	What year did his first film debut?
Human	What year did was Paul Verhoeven's first film debut?
BART _{CQR}	What year did <u>Steven Seagal's</u> first film debut?
RL-C	What year did <u>James Cameron's</u> first film debut? Flesh and Blood (1985)?
Original	Did he go into acting then?
Human	Did Coogan go into acting after college?
BART _{CQR}	Did <u>Charlie Chaplin</u> go into acting after <u>A Day's Pleasure</u> ?
RL-C	Did <u>Charlie Chaplin</u> go into acting after college?
Original	Did they do a second album?
Human	Did Gerry Mulligan and Chet Baker's quartet do a second album?
BART _{CQR}	Did <u>Pacific Jazz</u> do a second album?
RL-C	Did <u>Pacific Jazz</u> do a second album after PJ-8 ?

Table 8: Examples question rewrites where both RL-C and BART_{CQR} make mistakes.

Article:

The Seekers were offered a twelve-month position as on-board entertainment on the Sitmar Line passenger cruise ship Fairsky in March 1964. In May, they travelled to the U.K. and had intended to return to Australia after staying ten weeks, but upon arrival they were offered work by a London booking agency, the Grade Organisation. They signed there with World Record Club and issued a single, "Myra", co-written by the group. The group regularly appeared on a British TV show series, "Call in on Carroll", hosted by Ronnie Carroll. After filling in on a bill headlined by folk singer Dusty Springfield, they met her brother, songwriter and producer Tom Springfield, who had experience with writing folk-pop material and lyrics/tunes with the siblings' earlier group The Springfields. He penned "I'll Never Find Another You", which they recorded in November 1964. It was released by EMI Records, on their Columbia Graphophone Company (Columbia) label, in December and was championed by the offshore radio station "Radio Caroline" which frequently played and promoted their music. Despite the fact that the group had not signed a contract with EMI, the single reached the U.K. "Top 50" and began selling well. In February 1965, it reached No.1 in the U.K. and Australia, and No.4 in the United States where it was released on EMI's Capitol Records label. "I'll Never Find Another You" was the seventh biggest-selling single in Britain for 1965 though their own "The Carnival Is Over", released later in the year, would eventually eclipse it - and went on to sell 1.75 million copies worldwide. The Seekers were the first Australian pop group to have a "Top 5" hit in all three countries - Australia, U.K. and U.S.A. Australian music historian, Ian McFarlane described their style as "concentrated on a bright, uptempo sound, although they were too pop oriented to be considered strictly folk and too folk to be rock." The distinctive soprano voice of Durham, the group's vocal harmonies and memorable songs encouraged the British media, including the national broadcasting agency on radio and television, the BBC, to give them exposure, allowing them to appeal to a broad cross-section of the young British folk, pop and rock music audience.

Previous Conversation:

TURN NO.	QUESTION	CHATBOT ANSWER
Turn 1	What was their first album?	I don't know.
Turn 2	Does the article mention anything about that first album?	I don't know.
Turn 3	So why did you say not to follow up .(Is there anything additional about the album?	I don't know.
Turn 4	Who discovered the band?	EMI.
Turn 5	Is there any other important data about the bands discovery?	the single reached the U.K. "Top 50" and began selling well.

New question:

What was the single?

Alternative question 1	Alternative question 2
What was the Seekers single?	What was the single that reached the U.K. "Top 50" and began selling well?

Which of the alternative rephrased question is better ? (required)

Alternative question 1 is better

Alternative question 2 is better

Neither of the alternative is good

Both are good

Figure 3: Interface on Appen.

Overview

A ChatBot is a voice assistant like Alexa or Siri. In this task, we need your choice about which assistant understands the human questions better. Imagine you are reading an article. After reading, you have several questions about this article. You want to chat with the ChatBot to get the answers.

For each annotation task, imagine that you are in the middle of a conversation, where you already had the article, asked several questions and obtained some answers from ChatBot. These are the context of the conversation. **Assuming now you are asking a new question.** This new question might not contain enough information (e.g. Was it popular) for ChatBot to answer. For this, we also provide two alternative rephrased questions that might help ChatBot (e.g. Was the song popular). You are asked to select a better alternative from the given two.

Steps

1. Read **previous conversation**(1 min)
 - it is important for you to understand the new question.
2. Read and make sure you understand the **new question** that is shown as a continuation of this conversation.
 - The next question may contain references to concepts mentioned previously in the conversation that are not mentioned in the question itself.
3. Read and comprehend the shown 2 **alternative questions** .
4. (*optional*) Read the **article** this conversation is about (about 2mins) in case you:
 1. do not understand the **new question** even after reading previous **conversation**;
 2. are not sure whether the **alternative questions** are correct (both alternative questions can be bad).
5. Pick one of the **judgments** by comparing the **alternative questions** with the **conversation context**, the **new question**:
 1. **Alternative question 1 is better**: in case **Option 1** the first alternative question is better than **Option 2** and can be seen as a continuation of the previous conversation.
 2. **Alternative question 2 is better**: in case **Option 2** the second alternative question is better than **Option 1** and can be seen as a continuation of the previous conversation.
 3. **Neither of the alternative is good**: if none of the above criteria are met, that is, both rephrased questions cannot be seen as a continuation of the previous conversation, are not related to the new **question**, and are not about the **article** of interest.
 4. **Both are good**: in case both **Option 1** and **Option 2** are good alternative rephrases of the **new question** and are a continuation of the previous conversation.

Rules & Tips

To evaluate "*what is a good alternative question?*". Please considering the following metrics:

1. The question is *human readable* and *coherent*
2. The question provides more accurate information than the other alternative. For example, given two alternatives to the original **new question** "*Was it popular ?*":
 - Alternative 1: "*Was the song popular?*"
 - Alternative 2: "*Was the song by Winans popular?*"
 - Alternative 2 is better since it provides more information about the author of the song.

Figure 4: Instruction for annotators.

Examples

Examples 1

Article:

Winans recorded his second album *Hurt No More* in 2001, 2002, and 2003 in between working with other artists. The album is based on stories of love and betrayal. The first single "I Don't Wanna Know" was based on a sample of the Fugees' 1996 hit single "Ready or Not", which itself was based on a slowed-down sample of the instrumental track "Boadicea" by Enya from her 1987 self-titled album. Enya and her representatives became angry when Winans did not seek her approval for the sample, as he was unaware that the Fugees sample he had used had itself been a sample. So, a compromise was reached to credit the single as "Mario Winans featuring P. Diddy and Enya". The song features a rap by P. Diddy. The song was released as the single in early 2004 and became a worldwide hit, reaching number 1 in Germany and the United Kingdom and on the rhythmic top 40 radio chart in the United States; to number 2 on the Billboard Hot 100 and the US R&B/Hip-Hop singles chart; and to number 3 on both the Australian and Norwegian singles charts. It also reached the top ten on a composite European singles chart. *Hurt No More* was released on April 20, 2004 in the US and by early June 2004 had reached number 1 on the Billboard R&B/Hip-Hop Albums chart, number 2 on the Billboard 200, and number 3 on the UK Albums Chart. The album's second single, "Never Really Was", used a sample of the orchestrated beginning of Madonna's 1986 hit, "Papa Don't Preach", as its background music. However, the song failed to chart in America. A remix version was released as a single featuring rapper Lil Flip. The song "This Is the Thanks I Get" was stated to be the album's third and final single but no official release was set. The song went on to peak on the Billboard Bubbling Under R&B chart at number 12. CANNOTANSWER

Previous Conversation:

TURN NO.	QUESTION	CHATBOT ANSWER
Turn 1	what was hurt no more	Winans recorded his second album <i>Hurt No More</i> in 2001, 2002, and 2003
Turn 2	was it a success	by early June 2004 had reached number 1 on the Billboard R&B/Hip-Hop Albums chart,
Turn 3	did he win any awards for it	I don't know.
Turn 4	where there any popular singles off of it	The first single "I Don't Wanna Know" was based on a sample of the Fugees' 1996 hit single "Ready or Not"

New question: was it popular

ALTERNATIVE QUESTION 1	ALTERNATIVE QUESTION 2
was the single "I Don't Wanna Know" popular?	was "I Don't Wanna Know" by Winans popular?

Which of the alternative rephrased question is better ?

- Alternative question 1 is better
- Alternative question 2 is better
- Neither of the alternative is good
- Both are good

You select the system 2 since the alternative question 2 gives more detail about the author of the song.

Figure 5: An annotation example in the instruction.

Improving Large-Scale Conversational Assistants using Model Interpretation based Training Sample Selection

Stefan Schroedl Manoj Kumar Kiana Hajebi Morteza Ziyadi
Sriram Venkathapaty Anil Ramakrishna Rahul Gupta Pradeep Natarajan

Amazon Alexa AI, USA

{schroedl, abithm, hajkiana, mziyadi, vesriram, aniramak, gupra, natarap}@amazon.com

Abstract

Natural language understanding (NLU) models are a core component of large-scale conversational assistants. Collecting training data for these models through manual annotations is slow and expensive that impedes the pace of model improvement. We present a three stage approach to address this challenge: First, we identify a large set of relatively infrequent utterances from live traffic where the users implicitly communicated satisfaction with a response (such as by not interrupting), along with the existing model outputs as candidate annotations. Second, we identify a small subset of these utterances using *Integrated Gradients* based importance scores computed with the current models. Finally, we augment our training sets with these utterances and retrain our models. We demonstrate the effectiveness of our approach in a large-scale conversational assistant, processing billions of utterances every week. By augmenting our training set with just 0.05% more utterances through our approach, we observe statistically significant improvements for infrequent tail utterances: a 0.45% reduction in semantic error rate (SemER) in offline experiments, and a 1.23% reduction in defect rates in online A/B tests.

1 Introduction

Large-scale, voice-based conversational assistants such as Alexa, Siri, Google Assistant and Cortana process each utterance through a multi-stage pipeline that includes wakeword detection, automatic speech recognition (ASR), natural language understanding (NLU), entity resolution, and text-to-speech. This is a well-understood sequence (Sarikaya, 2017) and each of these steps leverage multiple machine learning models. The NLU system is often modularized into a number of *domains* that handle distinct classes of utterances such as Music, Weather, etc. (Su et al., 2018). The assistant system comprises models for *domain classification*

(DC), *intent classification* (IC), and *named entity recognition* (NER).

A key challenge in building, extending and maintaining such a system is that the underlying models need annotated training data. Collecting large volumes of such data through manual labeling is expensive and does not scale. Our work aims at improving the efficiency of this process. In contrast to previous approaches which identify utterances with defective responses, we instead focus on identifying cases that were processed successfully by the conversational assistant, and automatically retraining models with the additional data. However, this introduces two challenges. First, the vast majority of utterances are already processed correctly by the deployed system, resulting in an overwhelmingly large set of augmentation candidates. Secondly, implicit signals for satisfaction are noisy, as users might frequently ignore incorrect responses without making the effort to reformulate their query or provide a response that reflects dissatisfaction with the experience. Thus, simply adding all utterances from the full candidate pool (potentially billions/week) is infeasible and might actually degrade performance due to noise. We present a novel approach to address this based on *Integrated Gradients* (IG) (Sundararajan et al., 2017), a technique for understanding model behavior through feature importance scores. We propose a *sample importance score* that aggregates word scores and ranks the utterances in our initial candidate set, followed by training set augmentation with a small fraction of the top utterances.

Our experiments on live traffic data from a large scale conversational assistant indicate that retraining models with training sets, augmented by as little as 0.05% in size, produces a statistically significant (p -value < 0.05) improvement in semantic error rate (SemER) in offline test sets – 0.27% overall and 0.45% on a more challenging set of less frequent "tail" utterances. In online A/B tests, we

observe a 1.23% and 0.96% improvement in defect rates for all and tail utterances, respectively. In contrast, simply adding *all* utterances from our initial candidate set *degrades* SemER by 1.74% and 3.13% on the full and tail data sets, respectively. Finally, we demonstrate the repeatability and generalizability of our approach on public benchmark data sets. Despite the lack of label noise, we see small but consistent accuracy gains of 0.21% resp. 0.65% on the Snips and AGNews data sets.

2 Related Work

Several approaches have been proposed recently to use *distant* or *weak supervision* to address sparsity of labeled data (see e.g. the survey in (Hedderich et al., 2020)). A number of works identify utterances with processing errors through offline analysis (Sethi et al., 2021; Gupta et al., 2021; Chada et al., 2021; Khaziev et al., 2022). These approaches however still need human annotation in an active learning loop to improve production models. Query rewriting based approaches (Pon-usamy et al., 2020; Sodhi et al., 2021; Su et al., 2019; Hao et al., 2020) aim to address this limitation and enable self-learning without the need for human annotation. They detect instances where a user reformulates a query due to an unsatisfactory response and learn to map the failed utterance to a subsequent successful one. However, such approaches do not generalize to other utterance shapes. Falke et al. (2020) leverage user paraphrasing behavior in dialog systems to automatically collect annotations for long-tail utterances. Moerchen et al. (2020) present an approach where implicit negative feedback from the user is used to train a re-ranker that is then applied to pick correct annotations for under-performing utterances.

A range of post-hoc *model interpretability methods* for machine learned models has been developed in recent years (see e.g., (Madsen et al., 2021; Sundararajan et al., 2017; Ribeiro et al., 2016; Lundberg and Lee, 2017)). Local black-box methods typically measure the influence of individual features of an input example (e.g., individual words in a sentence) on the output prediction. Other techniques aim to score complete examples with respect to prototypicality (Carlini et al., 2019), influence on test predictions (Garima et al., 2020), and difficulty (Agarwal et al., 2022). Our word-occurrence based approach can be seen as a computationally scalable linear approximation to such measures.

Bhatt et al. (2019) conducted a survey on how organizations use model interpretability in practice. They identified model debugging as one of the primary uses of model explainability, seeking explicit human feedback on gathering more data for improving model performance.

Our approach differs from previous work as follows:

- There has been no prior work on the use of model interpretability in the context of data augmentation (though inversely, Chen and Ji (2019) proposed data augmentation to improve model explainability).
- Instead of detecting failed user interactions, we focus on utterances with implicit *positive* feedback.
- We leverage interpretability techniques in an automated way, without the need for human inspection.

3 Implicit User Feedback based Data Augmentation

NLU services which cater to a large number of users such as voice controlled agents typically collect implicit feedback metrics for each interaction. As a simplistic example, the absence of any negative feedback from the user to the agent’s action (no interruption, no repetition, etc.) can suggest that the agent successfully served the user’s request. In this paper, we propose a mechanism that relies on successful user interactions to identify additional data for building NLU models.

Oftentimes, an unsuccessful action from our virtual agent is followed by the user rephrasing their request. If the rephrase is successfully served by the agent, then this indicates that the NLU hypothesis for the rephrased request is likely correct. We believe that a correctly served rephrased turn is a stronger positive feedback when compared to a single-turn interaction with an implicit positive feedback (i.e no negative feedback from the user). We create a new training sample using the ASR transcript of the rephrased user request and the NLU hypothesis. We call this data set *weak signal labeled (WSL)* data since we rely on weak supervision from the user to obtain NLU labels. We score these utterances using integrated gradient technique as described in Section 4 before using them as additional data source for building NLU models.

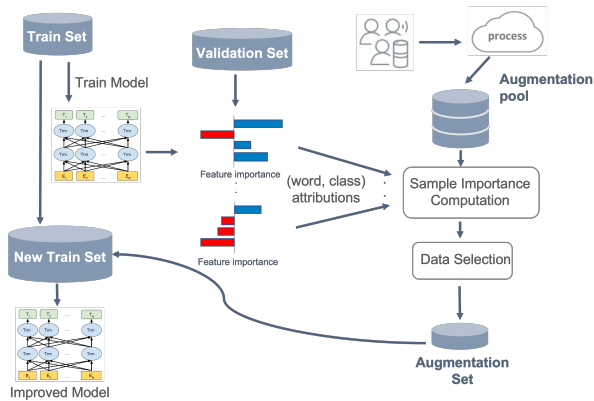


Figure 1: Model improvement based on feature attribution.

4 Importance Score Computation

We describe our approach for scoring utterances according to their importance to the performance of a classification model (e.g., a domain or intent classifier). Let T be the original training data, A be a pool of augmentation data, V be a validation data set, and Y be the test set. The model trained on T typically makes some mistakes on V . The objective is to use word attribution scores computed with a local black-box interpretability technique (Sundararajan et al., 2017; Ribeiro et al., 2016) on V to score utterances in A ; then, by adding some of them to T , we hope to train a model that is more robust against failures on Y that are similar to those observed on V . Thus, our approach can be roughly subdivided into three steps:

1. Calculation of an attribution score for each word in a misclassified utterance in V (with respect to the target and/or predicted class).
2. Aggregation of word scores over all instances.
3. Scoring of utterances in A based on the occurrences of important words.

See Figure 1 for a high level flow chart of our approach. We describe the details of each of these steps in the following sections.

4.1 Model interpretability

In this paper, we conduct experiments using the *Integrated Gradients* (Sundararajan et al., 2017) method. It is a local interpretation technique that addresses the problem of attributing a prediction of a deep network to its input features. Our approach is not restricted to it and it could be replaced with other methods such as LIME (Ribeiro et al., 2016) or SHAP (Lundberg and Lee, 2017). However, integrated gradients has several advantages:

word	True	Pred	T-Attr	P-Attr
tell	Books	Information	-0.61	+3.29
us	Books	Information	-1.01	+0.64
a	Books	Information	+0.37	+0.93
bedtime	Books	Information	-2.85	+3.27
story	Books	Information	+7.82	-3.80

Table 1: Feature attributions for true and predicted classes.

- It is scalable to large volumes of data.
- Its computed attributions are deterministic.
- It satisfies the desirable axioms of *linearity*, *implementation invariance*, and *sensitivity* (Sundararajan et al., 2017), which facilitate comparability of attributions across features and instances.

Integrated gradients require a non-informative baseline input. In the context of text processing, a natural choice is a sequence formed of padding tokens of the same length as the input. We interpret the words of an utterance as the features to be attributed, by averaging over token embedding vectors. Our implementation makes use of the PyTorch Captum package (Kokhlikyan et al., 2020).

Table 1 illustrates an example utterance in the validation set of the domain classifier along with the feature attributions of the words towards the true class (Books) as well as the predicted class (Information). We can see that the word `story` has positively influenced the model towards the true class but was not able to influence enough to make a right prediction. The word `tell` has positively influenced the utterance towards an incorrect prediction, while the word `bedtime` has negative influence towards the true class and high positive influence towards the predicted class. The objective, therefore, is to alter the training data so that the words `tell` and `bedtime` become more strongly associated with the class Books, especially in the context of the anchor word `story`.

4.2 Aggregation of word scores

The interpretability method produces an attribution mapping $\rho: (u, w, c, M) \rightarrow \mathbb{R}$, where u is an utterance in the validation set, w is a word in u , c is the class label, and M is the interpreted model.

Let the aggregated word scores be

$$g(w, c) = \sum_{(u,c) \in V, c' \in C} \max(0, -\rho(u, w, c', M) \cdot \delta(c, c', M)) \quad (1)$$

The function δ indicates the direction of the influence gap based on the true label of the utterance and the model prediction. Negative attributions with respect to the true class ($c = c'$), and positive attribution towards wrongly predicted classes ($c \neq c' = M(u)$) are summed over the validation set. Our objective is to enrich the training set with examples for the true class containing these words.

$$\delta(c, c', M) = \begin{cases} 1 & c = c' \\ -1 & c \neq c' = M(u) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

4.3 Score an utterance using word attributions

Let $G = \{(w, c)\}$ be the set of all word attributions computed according to Eqn. 1. The most straightforward way of computing the score of utterance u of class c is the *greedy method* of summing up the importance scores of all word occurrences:

$$h(u, c) = \sum \{g(w, c) \mid w \in u, (w, c) \in G\} \quad (3)$$

Then, we select the top n utterances from A according to this score.

The greedy method has the drawback that it can become too narrowly focused on just a few high-scoring words, thus leading to heavily imbalanced augmentation data sets. We introduce a *diversity method* as a remedy. The idea is to incrementally discount a score pair after selecting an utterance containing it. One simple way of doing so is by dividing the word importance by the number of such utterances, as outlined in Algorithm 1.

5 Experimental Setup

We first present initial results of our approach on the open source intent classification (IC) data sets (Snips (Coucke et al., 2018) and AGNews (Del Corso et al., 2005)), and then demonstrate the impact of our approach on a joint intent classification and named entity recognition (IC-NER) task on live traffic of a commercial conversational assistant.

Algorithm 1 Diversity method for utterance scoring.

```

function SELECT_DIVERSITY( $n, G, A$ )
  for all  $(w, c) \in G$  do
     $n(w, c) \leftarrow 1$ 
  end for
   $S \leftarrow \{\}$ 
  for  $i \leftarrow 1, \dots, n_{augment}$  do
    for all  $(u, c) \in A \setminus S$  do
       $h(u, c) \leftarrow \sum \{g(w, c) / n(w, c) \mid$ 
         $w \in u, (w, c) \in G\}$ 
    end for
     $(u', c') \leftarrow \arg \max_{u, c} h(u, c)$ 
     $A \leftarrow A \cup \{(u', c')\}$ 
    for all  $w' \in u', (w', u') \in G$  do
       $n(w', c') \leftarrow n(w', c') + 1$ 
    end for
  end for
  return  $S$ 
end function

```

5.1 Data sets

5.1.1 Open source data sets

Snips (Coucke et al., 2018) is a natural language understanding benchmark data set of over 16 000 crowdsourced queries distributed among 7 NLU intents. It is pre-split into a training set (13 084 utterances), validation set, and test set (700 utterances each with 100 queries per intent).

With **AGNews** (Del Corso et al., 2005), we chose a data set with a slightly different but related task (news topic classification), due to its sufficient size. It contains 4 classes each containing 30 000 training samples and 1 900 testing samples. The total number of training and testing data is 120 000 and 7 600, respectively. We apply stratified random sampling to subdivide the training further for a validation set of 7 600 instances.

Apart from the usual partitions of data into train, validation and test sets, our experiments consider a further sub-partition of the train set into base set and augmentation set. The base set is a randomly down-sampled version of the full train set to a desired target size (e.g., 50% of the data). The rest of the training data is the augmentation set from which additional samples are selected. The validation set is used for computing the feature attributions.

5.1.2 Proprietary data from conversational assistant

In these experiments, we work with logs of user interactions with our conversational assistant. This data is prepared in accordance with our general strict privacy protection procedures. All production data is de-identified so that it is not personally identifiable.

We evaluate our approach on the utterances from a random partition of live traffic as well as on a set of low frequency *tail* utterances. Tail utterances constitute a significant portion of the overall traffic, and measure the statistical model’s ability to generalize to a wide gamut of real-world utterances of users. Improving a machine-learned model’s performance on rare utterances is of increasing interest among industrial and academic applications, as defects in frequently recurring head utterances can often readily be addressed using rule based systems. We compare models in terms of offline NLU performance, but also run live traffic A/B experiments to directly measure the user impact.

Our weak signal labelled data stems from unique utterances with implicit positive user feedback across all domains over a period of time. For improved precision, we remove utterances with ASR confidence scores below an empirically determined threshold. We rank the utterances within each domain using the scores obtained using interpretability methods, greedy and diversity, as described in Section 4.1. The WSL data set thus prepared represents $\approx 8.5\%$ of the total training data size. For each domain, we rank WSL utterances in the order of decreasing relevance: we favor utterances which are likely to influence the domain classification model predictions the most. We select a small fraction of the most influential utterances (0.05% of the training data) and fine-tuned IC-NER models. The amount of data that can be augmented is limited by engineering constraints (e.g., model build times, storage capacity), hence the interpretability-based scores are useful to identify an optimum subset of utterances that provide the most utility.

5.2 Models

On the live traffic data set, we use a joint IC-NER model with a distilled version of BERT encoder pretrained with MLM objective on a combination of public and internal data sets. The total parameter count of the encoder is $17M$. We use a sentence-piece tokenizer of size $150K$ sub-word units. For

each of IC and NER tasks, the model uses feed-forward layers of hidden size 256 followed by softmax layer. We train with early stopping, up to 10 epochs, at a learning rate of $5e-5$ and a batch size of 32. For the simpler public domain datasets, we fine-tune the DistilBERT model from Huggingface¹ ($65M$ parameters) as an intent classifier.

6 Experiments and Results

6.1 Snips

For the experiments with data augmentation on Snips, we use the model trained on $\approx 50\%$ of the training data (T) – constituting 8 192 out of 13 084 samples – as the base model. The remaining data is considered the augmentation set (A). We investigate the accuracy impact of augmenting a small fraction of examples to the training set with our importance scoring approach. Specifically, we explore the greedy and diversity methods as explained in Section 4.3, with a set of 82 ($\approx 1\%$) augmentation utterances selected from A . As shown in Table 2, the diversity method improves performance to 0.976 compared to the baseline accuracy of 0.974, while a model trained with the full augmentation data set has 0.978 accuracy.

6.2 AGNews

For our interpretability experiments on the AG-News data set, we choose a base model trained on $\approx 25\%$ of data set, achieving an accuracy of 0.923. According to Table 2, the diversity method achieves a test accuracy of 0.929, an improvement over the baseline by 0.6%.

6.3 Data augmentation with weak signal data

We evaluate the utility of WSL data augmentation using model interpretability scores for NLU models of the conversational assistant. We build a domain-specific IC-NER model using the same training data as in the production setting. All IC-NER models share a common encoder as described in Section 5. The output dimension for each model depends on the number of intents and slot labels for each domain. We use similar training parameters (epochs, learning rate, optimizer, etc.) as production settings and defer any hyper-parameter tuning experiments for future work. We refer to this model as *baseline*.

For each domain, we build a second model (*WSL*) using the same architecture and training parameters as the baseline model. We augment all

¹<https://huggingface.co/distilbert-base-uncased>

data set	Full size	Baseline size	Modification size	Baseline accuracy	Augmentation		Full accuracy
					Diversity	Greedy	
Snips	13,084	8,192	82	0.974	0.976	0.971	0.978
AGNews	112,400	32,768	64	0.923	0.929	0.926	0.942

Table 2: Accuracy of intent classification on Snips and topic classification on AGNews data sets, comparing different approaches with random selection baseline. Each number is the average over 5 runs with different seeds.

Table 3: Relative semantic error rates (SemER) for IC-NER models trained on all WSL data (WSL), and WSL data filtered with interpretability-based scores, greedy and diversity, (WSL-IG). All metrics are reported relative to baseline model ($p < 0.05^*$).

Model	All	Tail
Baseline	0%	0%
WSL (no filtering)	1.74%*	3.13%*
WSL-IG (greedy)	-0.27%	-0.45%*
WSL-IG (diversity)	-0.13%	-0.33%*

the WSL data described in Section 3 to the training data, before applying importance scores for utterance selection. Finally, we build a third model (WSL-IG) which uses interpretability-based scores to select the most relevant utterances.

We report the IC-NER task performance using weighted semantic error rate (SemER; (Makhoul et al., 1999; Su et al., 2018)) metric. We construct a label sequence for each utterance by concatenating the intent and slots (in order). Given the total count of erroneous insertion (I), erroneous deletions (D), substitutions (S) and correct labels (C), SemER is computed as: $S = \frac{(I+D+S)}{(C+D+S)}$. In Table 3, we report the weighted mean of SemER relative to the baseline model and weighted by the domain’s test utterance count. We compare the baseline and proposed models on two test sets: (i) *All* contains user queries from the entire traffic; (ii) *Tail* contains only low-frequency requests.

From Table 3, we notice that the interpretability-based filtering plays an important role in improving the semantic error rate on both test sets. SemER reductions obtained with WSL-IG models are significant at $p < 0.05$ on the tail test set. The magnitude of SemER improvement is higher on the Tail test set, which is likely due to the similar nature of WSL utterances (sourced from low-frequency traffic). Interestingly, WSL models which are built using the largest training data sizes are significantly worse than the baseline, illustrating the noisy nature of implicit user feedback. In contrast to our Snips and AGNews results, the greedy method per-

Table 4: Relative defect rate from online A/B experiment comparing NLU models built with WSL data. The defect rates are reported for low-frequency utterances (*Tail*) and all utterances (*All*) relative to the control model ($p < 0.05^*$).

	Overall	General	Information
All	-1.23%*	-0.27%	-1.04%*
Tail	-0.96%*	-1.32%*	-1.64%*

forms better than diversity – possibly due to the much larger training size, and suggesting a more diverse range of defect patterns.

We followed up with an online A/B experiment on our voice-controlled agent to test the impact of WSL data on live traffic. We experiment with two domains: *General* which serves generic requests such as turn on device, change volume, etc. and *Information*, which serves general knowledge related requests. For both domains, user requests on the treatment group were served by NLU models trained with additional WSL data which were filtered using interpretability-based scores, (WSL-IG). We measure the outcome of the A/B experiment using an internal business metric (referred to as defect rate) which estimates whether the agent was successful in serving the user’s request. Success is estimated based on the user’s perception following the agent’s response. For example, it is likely that the agent has misinterpreted the user’s request when the user rephrases/repeats their request or the agent communicates that it cannot serve the user’s request: "sorry I don’t know the answer to that". We present the relative change in defect rate on both low-frequency utterances (*Tail*) and all utterances (*All*). From Table 4, we observe improvements in the defect rate across both deployed domains and the overall traffic. For both domains, while defect reductions are observed on both All and Tail test sets improvements in the latter are more noticeable, which demonstrate the utility of interpretability-based filtering of implicit customer feedback for NLU model building.

7 Conclusions and Future Work

A key challenge in building state-of-the-art deep learning models is the cost and effort involved in obtaining large volumes of manually labeled data. Our work is part of a line of investigations into leveraging unlabeled or weakly supervised data at scale. We extract large amounts of user dialogs with a conversational assistant which are deemed successful according to implicit feedback. However, it is not sufficient to add all such examples indiscriminately to the training data – doing so does not improve the model, nor is it computationally scalable. We show how to leverage model interpretability techniques to prioritize the most important instances that should be added to the training set. Our approach leads to statistically significant error rate reductions of our live system. We also demonstrate transferability on public NLU data sets, Snips and AGNews.

In the future, we will apply our approach to other challenging public data sets which suffer from significant label noise and ambiguity. We will investigate other types of data set modifications, such as removal and replacement of examples.

References

- Chirag Agarwal, Daniel D’souza, and Sara Hooker. 2022. Estimating example difficulty using variance of gradients. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10368–10378.
- Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2019. [Explainable machine learning in deployment](#). *CoRR*, abs/1909.06342.
- N Carlini, U Erlingsson, and N Papernot. 2019. [Prototypical examples in deep learning: Metrics, characteristics, and utility](#). *arXiv*.
- Rakesh Chada, Pradeep Natarajan, Darshan Fofadiya, and Prathap Ramachandra. 2021. [Error detection in large-scale natural language understanding systems using transformer models](#). In *ACL-IJCNLP 2021*.
- Hanjie Chen and Yangfeng Ji. 2019. [Improving the interpretability of neural sentiment classifiers via data augmentation](#). *CoRR*, abs/1909.04225.
- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Calta-girone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.
- Gianna M Del Corso, Antonio Gulli, and Francesco Romani. 2005. Ranking a stream of news. In *Proceedings of the 14th international conference on World Wide Web*, pages 97–106.
- Tobias Falke, Markus Boese, Daniil Sorokin, Caglar Tirkaz, and Patrick Lehnen. 2020. [Leveraging user paraphrasing behavior in dialog systems to automatically collect annotations for long-tail utterances](#). In *COLING 2020*.
- Garima, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. [Estimating training data influence by tracing gradient descent](#). In *Advances in Neural Information Processing Systems*, volume 2020-Decem.
- Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei (Edward) Guo. 2021. [Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems](#). In *KDD 2021 Workshop on Data-Efficient Machine Learning*.
- Jie Hao, Linfeng Song, Liwei Wang, Kun Xu, Zhaopeng Tu, and Dong Yu. 2020. Robust dialogue utterance rewriting as sequence tagging. *CoRR*, abs/2012.14535.
- Michael A Hedderich, Lukas Lange, Heike Adel, Jan-nik Strötgen, and Dietrich Klakow. 2020. A survey on recent approaches for natural language processing in low-resource scenarios. *arXiv preprint arXiv:2010.12309*.
- Rinat Khaziev, Usman Shahid, Tobias Röding, Rakesh Chada, Emir Kapanci, and Pradeep Natarajan. 2022. FPI: Failure point isolation in large-scale conversational assistants. In *NAACL-HLT*.
- Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, et al. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.
- Scott M Lundberg and Su-In Lee. 2017. [A unified approach to interpreting model predictions](#). In *Neural Information Processing Systems*.
- Andreas Madsen, Siva Reddy, and Sarath Chandar. 2021. Post-hoc interpretability for neural NLP: A survey. *arXiv preprint arXiv:2108.04840*.
- John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. 1999. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*, pages 249–252.
- Fabian Moerchen, Patrick Ernst, and Giovanni Zappella. 2020. [Personalizing natural-language understanding using multi-armed bandits and implicit feedback](#). In *CIKM 2020*.

- Pragaash Ponnusamy, Alireza Roshan-Ghias, Chenlei (Edward) Guo, and Ruhi Sarikaya. 2020. [Feedback-based self-learning in large-scale conversational AI agents](#). In *AAAI 2020*.
- Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why should I trust you?": Explaining the predictions of any classifier. *CoRR*, abs/1602.04938.
- Ruhi Sarikaya. 2017. The technology behind personal digital assistants: An overview of the system architecture and key components. *IEEE Signal Process. Mag.*, 34(1):67–81.
- Pooja Sethi, Denis Savenkov, Forough Arabshahi, Jack Goetz, Micaela Tolliver, Nicolas Scheffer, Ilknur Kabul, Yue Liu, and Ahmed Aly. 2021. AutoNLU: Detecting, root-causing, and fixing NLU model errors. *CoRR*, abs/2110.06384.
- Sukhdeep S. Sodhi, Ellie Ka In Chio, Ambarish Jash, Santiago Ontañón, Ajit Apte, Ankit Kumar, Ayooluwakunmi Jeje, Dima Kuzmin, Harry Fung, Heng-Tze Cheng, Jon Effrat, Tarush Bali, Nitin Jindal, Pei Cao, Sarvjeet Singh, Senqiang Zhou, Tameen Khan, Amol Wankhede, Moustafa Alzantot, Allen Wu, and Tushar Chandra. 2021. Mondegreen: A post-processing solution to speech recognition error correction for voice search queries. *CoRR*, abs/2105.09930.
- Chengwei Su, Rahul Gupta, Shankar Ananthakrishnan, and Spyros Matsoukas. 2018. A re-ranker scheme for integrating large scale NLU models. In *2018 IEEE Spoken Language Technology Workshop, SLT 2018, Athens, Greece, December 18-21, 2018*, pages 670–676. IEEE.
- Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance ReWriter. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 22–31, Florence, Italy. Association for Computational Linguistics.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. [Axiomatic attribution for deep networks](#). *CoRR*, abs/1703.01365.

Improving Precancerous Case Characterization via Transformer-based Ensemble Learning

Yizhen Zhong, Jiajie Xiao, Thomas Vetterli, Mahan Matin,
Ellen Loo, Jimmy Lin, Richard Bourgon, Ofer Shapira

Freenome, South San Francisco, CA

{yizhen.zhong, jiajie.xiao, thomas.vetterli, mahan.matin}@freenome.com

{ellen.loo, jimmy.lin, richard.bourgon, ofer.shapira}@freenome.com

Abstract

The application of natural language processing (NLP) to cancer pathology reports has been focused on detecting cancer cases, largely ignoring precancerous cases. Improving the characterization of precancerous adenomas assists in developing diagnostic tests for early cancer detection and prevention, especially for colorectal cancer (CRC). Here we developed transformer-based deep neural network NLP models to perform the CRC phenotyping, with the goal of extracting precancerous lesion attributes and distinguishing cancer and precancerous cases. We achieved 0.914 macro-F1 scores for classifying patients into negative, non-advanced adenoma, advanced adenoma and CRC. We further improved the performance to 0.923 using an ensemble of classifiers for cancer status classification and lesion size named entity recognition (NER). Our results demonstrated the potential of using NLP to leverage real-world health record data to facilitate the development of diagnostic tests for early cancer prevention.

1 Introduction

Cancer has been the second leading cause of death with more than 1,900k new cases and 600k cancer deaths in the United States in 2022 (Siegel et al., 2019). Among those, colorectal cancer (CRC) is the third most common cancer and the third leading cause of cancer death (Siegel et al., 2019). Detecting CRC at its early stage can dramatically improve clinical outcomes. The 5-year survival rate is 90% when colorectal cancer is identified at the localized stage compared to 73% and 17% survival rates at the regional or distant stage, respectively¹.

CRC progresses from asymptomatic non-advanced adenoma (NAA) to advanced adenoma (AA) and then to invasive carcinoma (Junca et al., 2020). AAs are adenomas characterized by villous

or tubulovillous histology, adenomas or sessile serrated lesions $\geq 10\text{mm}$, or high-grade dysplasia (Junca et al., 2020; Shaukat et al., 2021). AA indicates an intermediate or high risk for CRC (Lieberman et al., 2012) and requires CRC screening every three years (Lieberman et al., 2012). Recent economic studies suggest a test with increasing adenoma sensitivity in a blood-based CRC screening test can reduce CRC incidence and reduce mortality (Putcha et al., 2022a).

There is great interest in developing noninvasive diagnostic tests with high sensitivity and specificity for advanced adenoma and CRC screening (Imperiale et al., 2014; Putcha et al., 2022a). This development process requires biomarker discovery and clinical validation based on samples collected from large numbers of individuals whose colorectal cancer statuses are confirmed by colonoscopy (Putcha et al., 2022b). Correctly classifying the colorectal cancer statuses, namely negative (NEG), NAA, AA and CRC, requires expertise in distilling and interpreting tumor stage and histology information and size of precancerous adenoma from colonoscopy and pathology reports. Such nuanced annotations are typically not documented and collected in structured sections of electronic health records or standardized via International Classification of Diseases (ICD) codes (Raju et al., 2015, 2013). Therefore, extracting this information from colonoscopy and pathology reports and generating reliable CRC status classification has heavily relied on manual review by trained gastrointestinal pathologists. Such review is time-consuming, costly and difficult to scale.

To reduce the burden of manually annotating thousands to hundreds of thousands of pathology reports, and to facilitate the development of noninvasive diagnostic tools for colorectal cancer prevention, we investigated classical and advanced natural language processing (NLP) methods to automatically extract precancerous lesion information and

¹<https://www.cancer.org/cancer/colon-rectal-cancer/detection-diagnosis-staging/survival-rates.html>

determine CRC status (Figure 1). We developed transformer models to extract both categorical and numerical attributes from colonoscopy and pathology reports. Compared to Bag-of-Word (BoW) and convolutional neural network (CNN) models (see Data and methods in section 3), we achieved the best performance by fine-tuning the BioBERT model. Since lesion size is an important factor to distinguish between the AA and NAA classes (Appendix A.1, Winawer and Zauber (2002)), we developed an entity recognition model for lesion size extraction and improved its performance through transfer learning from a non-biomedical domain. We further improved the cancer status classification model performance by explicitly adding extracted lesion size through an ensemble model. Our study also addressed two challenges for NLP practice that are specific to the biomedical industry setting: annotation at the sentence level for numerical variable extraction is limited; and most clinical trial studies that enroll patients from various sites still receive health records in the scanned PDF format (Raju et al., 2015), creating challenges for precisely locating the diagnosis section in health records. Our research demonstrated that, along with domain knowledge-informed feature learning, fine-tuned advanced deep learning methods are able to achieve high accuracy in highly complex and nuanced disease phenotyping tasks, even with only several thousands of documents for model training.

2 Related Work

NLP methods have been applied to pathology reports to extract categorical attributes associated with cancer diagnosis. Yala et al. (2017) used machine learning methods with Bag-of-Words features to classify patients into breast cancer carcinoma and atypia categories. Adding clinical concepts from the Unified Medical Language System (UMLS) was shown to improve classification performance (Li and Martinez, 2010; Martinez and Li, 2011). Preston et al. (2022) used embedding vectors that are pre-trained in BERT-based (Devlin et al., 2018) models for tumor site, histology and TNM staging (T: tumor size/location; N: lymph node status; M: metastasis) classification from longitudinal reports and developed classifiers to detect cancer cases. Park et al. (2021) extracted cancer histology, site and surgical procedure from colon, lung and kidney cancer data. They demonstrated good performance by leveraging transfer learning

across cancer types and few-shot learning by accounting for semantic similarity. Other deep learning approaches such as hierarchical attention neural networks (Gao et al., 2018, 2019), multitask learning (Alawad et al., 2020), and graph convolutional networks (Wu et al., 2020) have been employed to extract cancer characteristics such as primary site and histological grade. However, these studies primarily focused on extracting categorical cancer characteristics that are routinely collected in cancer registries (Klein and Havener, 2011) and largely ignored numerical and precancerous attributes, which are critical for developing early cancer detection technology.

The extraction of numerical cancer attributes is challenging because the semantic context for numerical variables is mostly at the sentence level instead of document/patient level (Li and Martinez, 2010; Odisho et al., 2020). This creates a discrepancy between training objectives (sentence level) and output evaluation (patient level). To overcome this limitation, Li and Martinez (2010) first identified the sentences that contain the numerical values and extracted them through regular expression matching. AAIAbdulsalam et al. (2018) treated TNM staging extraction as a sequence labeling task with pattern matching and conditional random field techniques. Odisho et al. (2020) encoded tokens and their context words as bag-of- n -gram features and classified the token sequence for TNM staging and tumor volume extraction. In our work, we treated numerical lesion size extraction as a named entity recognition (NER) task and addressed the challenge of limited annotated sample size by transfer learning from models pre-trained in a non-biomedical domain.

Previous attempts to employ NLP methods to parse colonoscopy reports and linked pathology reports have aimed to characterize adenomas due to their importance in estimating colorectal cancer risk (Lee et al., 2019; Raju et al., 2013, 2015; Imler et al., 2013). The limitations of current studies are two fold: first, most studies still rely on rule-based systems such as Linguamatics (Lee et al., 2019) and cTAKes (Imler et al., 2013; Savova et al., 2010) for extracting adenomas through pattern-matching and dictionary look-up. Deriving rules can be time-consuming, require extensive domain knowledge, and likely results in overfitting to the development dataset and limited portability; second, NLP studies for colorectal cancer do not perform end-to-end

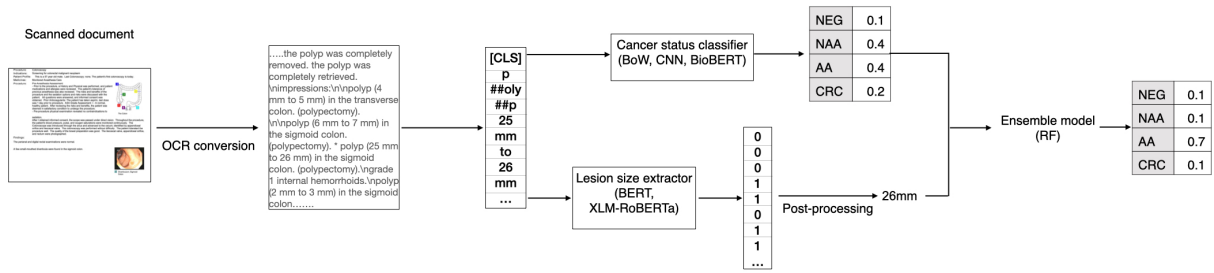


Figure 1: Overall framework

CRC phenotyping to classify patients into NEG, NAA, AA and CRC, which are of great interest to characterize CRC risk and prioritize patients for cancer screening and early cancer detection. Here we performed end-to-end precancerous and cancer status characterization with deep learning methods which promise to be more generalizable and efficient than rule-based approaches.

3 Data and methods

3.1 Dataset

In this study, we used health records from 3,068 patients collected as part of two studies from 68 collection sites. In some cases, multiple types of health records are associated with one patient, including colonoscopy, pathology, surgical pathology, and radiology reports. In total, there are 5,405 documents for all patients. Appendix Figure A1 shows the distribution of document numbers for each patient.

We split patients into train and test sets stratified by cancer status. To assess the generalizability of NLP models when applied to pathology reports collected in unseen sites, we used samples from independent collection sites for the train and test sets. There are 2,149 samples in the training set from 54 sites and 919 samples in the test set from 14 independent sites. Appendix Table A1 shows the sample count for each cancer status in the train and test sets.

3.2 Document and sentence level annotations

A certified pathologist reviewed and assigned patient-level labels for colorectal cancer status and lesion size. The detailed annotation criteria are described in Appendix A.1. For lesion size annotation, the pathologist first identified the index lesion, which is the most clinically significant lesion according to cancer status and lesion type. Then the size of the index lesion was used as the patient-

level lesion size annotation. We used zero as the lesion size for healthy samples with no identified lesions.

We generated sentence-level annotation for lesion size. We treated the lesion size named entity annotation as a binary label with tokens within the lesion size entity as 1 and tokens outside the lesion size entity as 0. We did not distinguish the start or end token of the named entity. We randomly selected 499 documents from 225 patients from the training set and 331 documents from 114 patients from the test set for NER model training and evaluation, respectively.

3.3 Data preprocessing

Since the reports are in scanned PDF format, we first digitized the reports with optical character recognition provided by the Google Vision API². The OCR algorithm outputs the recognized text and the coordinates of the bounding box for each text block.

We used fuzzy matching for words: "diagnosis," "finding," "impression," "diagnoses," "findings," "impressions," and "polyp" with text in each bounding box. This allowed us to identify sections important for diagnosis and ignore irrelevant sections to increase the signal-to-noise ratio. To allow for some error in bounding box identification, we retained texts within 10 bounding boxes and at most 100 words after the first matched bounding box. These keywords for fuzzy matching and window size were determined by an iterative manual inspection of reports in the training set.

For CNN and BoW, we concatenated documents corresponding to one patient into one text segment and padded the concatenated text segment to the max length. For the BERT models with a sequence length limit of 512 tokens, we split the text into segments with 10 overlapping tokens.

²<https://cloud.google.com/vision/docs/ocr>

3.4 Cancer status classification model

We built and tested three models, namely Bag-of-Words (BoW) (Zhong et al., 2018), CNN (Kim, 2019) and BioBERT (Lee et al., 2020), for cancer status classification. We split the 2,149 documents reserved for training into training and validation sets in a 9:1 ratio and selected the best model based on the validation macro-F1 score [See 3.7]. For BoW, *tf-idf* representation (Zhong et al., 2018), a term-frequency based featurization, was derived as input features for SVM models with linear, polynomial or radial basis function (RBF) kernels. We used unigram features and removed terms that appear in less than 10 documents.

For the CNN model, instead of doing hand-crafted feature engineering, 1D convolution kernels were learned to extract localized text patterns from pathology reports. The convolutional layer is followed by a max-pooling layer and a fully connected layer to classify colorectal cancer status.

For the BERT model, BioBERT (Lee et al., 2020), a pre-trained biomedical language representation, was employed and fine-tuned as follows to encode the pathology reports for cancer status classification. BioBERT was pre-trained based on BERT initiated weights with biomedical domain corpora (PubMed abstracts and PMC full-text articles) and has increased performance in biomedical text mining tasks including NER, relation extraction and question-answering. We added a fully connected layer after the [CLS] embedding vector for multiclass classification (Devlin et al., 2018). Because one patient can be associated with multiple documents or text segments but the cancer status label is annotated for each patient, this creates a multiple instance learning problem. We treated each patient as a bag and each text segment as an instance within the bag. We used max-pooling to get the largest softmax probability for each class across multiple text segments and renormalize with the softmax function to calculate cross-entropy loss per patient.

All these models were optimized through Bayesian hyperparameter tuning (Snoek et al., 2012; Shahriari et al., 2015) with early stopping from the Hyperband algorithm (Li et al., 2017). More details of the procedures can be found in the Appendix Table A3.

3.5 Lesion size extraction model

We treated the lesion size extraction as a NER task and compared two approaches. For direct fine-tuning, we used a pretrained BERT-base-uncased³ model and classified token embedding into binary labels where the positive label indicates the target named entity. We also fine-tuned a XLM_RoBERTa_base⁴ model.

Observing the similarity between lesion size vs. one of the annotated named entities (QUANTITY: Measurements, as of weight or distance) from the OntoNotes5 corpus (Weischedel et al., 2011), we used an XLM_RoBERTa⁵ model that was previously fine-tuned on the OntoNotes5 dataset for NER of QUANTITY. We then continued to fine-tune this model on the cancer pathology dataset for lesion size extraction to explore the benefit of transfer learning. Both direct fine-tuning and transfer learning models were trained and validated based on a 7:1 split of the sentence-level annotated documents. We selected the model with the best validation F1 score and evaluated its performance on the holdout test set. The hyperparameters can be found in Appendix Table A3.

3.6 Ensemble model

We built an ensemble model with BioBERT predicted probability for each class and binarized lesion size feature (lesion size ≥ 10 mm or not). For the model training, we used the binarized ground-truth lesion size. For model inference, we used the binarized NER-extracted lesion size. We trained a random forest model with 10 trees and max_depth=10 as the ensemble model on the training set and tested its performance on the validation and test sets for the cancer status classification task.

3.7 Metrics

For cancer status classification evaluation, we computed the precision, recall and F1 for each cancer status. We used the macro-F1= $\sum_i^n \frac{F_i}{n}$ for the overall model performance metric for multi-class classification (n classes).

For NER evaluation, we count consecutive positive labeled tokens as one named entity. We identified named entities derived from ground-truth labels (total ground-truth positive, TGP) and predicted labels (total predicted positive, TPP). We

³<https://huggingface.co/bert-base-uncased>

⁴<https://huggingface.co/xlm-roberta-base>

⁵<https://huggingface.co/asahi417/>

tner-xlm-roberta-base-uncased-ontonotes5

counted an exact match of starting and ending index of the ground-truth and predicted entities as true positive (TP). We calculated the precision as $\frac{TP}{TPP}$, recall as $\frac{TP}{TGP}$ and F1 score for the named entity recognition task.

4 Results

4.1 Cancer status classification model performance

We treated the cancer status (CRC, AA, NAA and NEG) extraction as a document classification problem and trained BoW, CNN and BioBERT models. All models achieved over 0.8 macro-F1 scores, with the BioBERT model outperforming BoW and CNN (Table 1). In particular, all models including BoW classified CRC and NEG with high accuracy (> 0.9 F1 score) and AA and NAA with lower accuracy (< 0.8 F1 score). This suggests that unigram features are sensitive for classifying cancer and healthy patients from pathology reports but less sensitive for differentiating precancerous patients.

The CNN model (NAA F1=0.842, AA F1=0.773) improved NAA and AA performance compared to BoW (NAA F1=0.706, AA F1=0.758). This suggests that the larger kernels used in CNN improve the capture of semantics for precancerous classes compared with unigram features. The BioBERT model further improved AA and NAA performance (NAA F1=0.888, AA F1=0.833). As the model complexity and its ability to capture long-range interaction increases, the model performed better. Although the number of training samples was limited, the more complex models appear to be more generalizable.

4.2 Lesion size extraction model performance

We performed an error analysis on the training and validation data to identify the source of incorrect predictions (Table 2, Appendix Table A2) for the BioBERT model. We found misclassifications were usually confusion between AA and NAA. 68.0% (17/25) of incorrect predictions for NAAs were classified as AA, and 65.0% (13/20) of incorrect predictions for AAs were classified as NAAs. Since lesion size is an important factor to distinguish between the AA and NAA classes (Appendix A.1), we proposed to explicitly add lesion size as an additional feature to improve the BERT-based cancer status classification model.

The direct fine-tuning of the BERT model for lesion size NER had low performance, potentially

due to the small sample size for sentence-level annotation (test F1 score=0.202, precision=0.159 and recall=0.273, Table 3). We then evaluated the transfer learning approach, using an XLM_RoBERTa model that had been fine-tuned on the OntoNotes dataset for QUANTITY extraction. Directly applying this model to the cancer pathology dataset to extract lesion size led to an increased F1 (0.508) score, with high recall (0.703) and low precision (0.398). We next continued to train this model on the cancer pathology dataset to perform lesion size extraction. Interestingly, additional fine-tuning substantially improved the performance (F1=0.757, precision=0.761 and recall=0.753), especially the precision. This suggests that transfer learning using models fine-tuned on tasks outside the biomedical domain can substantially improve domain-specific NLP performance, even with a relatively small sample size.

4.3 Ensemble model improves cancer status classification

We then assessed the effect of explicitly adding lesion size as an additional feature to classify cancer status. The ensemble model which combined BioBERT predicted probabilities and binarized lesion size (≥ 10 mm or not) improved NAA performance from 0.888 to 0.894 and AA performance from 0.833 to 0.854 while maintaining the already high performance for the NEG and CRC classes (Table 1). The macro-F1 was 0.923 for the ensemble model compared to 0.914 for the BioBERT model alone. This suggests that explicitly adding features that are informed by domain knowledge can improve classification performance compared to fine-tuned transformer models alone. This approach may be particularly beneficial for applications in which training data are limited.

4.4 Integrated gradient analysis

To investigate which features are most important for BioBERT model performance, we performed integrated gradient analysis, which computes attribution scores that measure feature importance with respect to the classification prediction (Sundararajan et al., 2017). We calculated attribution with respect to the input embedding vector. We performed integrated gradient analysis for a random subset of 534 NEGs, 197 NAAs, 168 AAs and 80 CRCs and calculated the averaged feature attributions across documents for each class (Figure 2). High-scoring tokens related to CRC classifica-

Model	NEG F1 (n=515)	NAA F1 (n=183)	AA F1 (n=146)	CRC F1 (n=75)	Macro-F1
BoW	0.941	0.706	0.758	0.952	0.839
CNN	0.971	0.842	0.773	0.947	0.883
BioBERT	0.972	0.888	0.833	0.962	0.914
BioBERT+Lesion Size	0.965	0.894	0.854	0.980	0.923

Table 1: Performance of cancer status classifiers as measured by the test F1 scores for each class and their macro-averages.

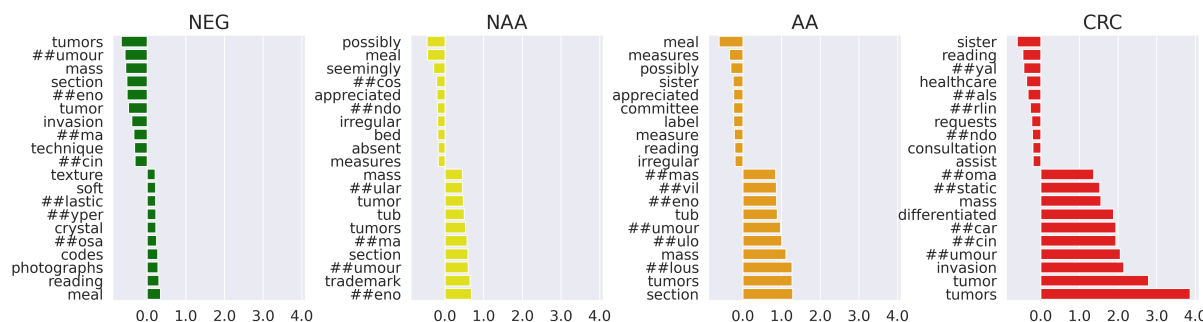


Figure 2: Attribution score. Tokens with the largest integrated gradient analysis attribution score for each cancer status. (Top 10 positive and negative attribution scores.)

True Label	Predicted Label	Predicted Label				Total
		NEG	NAA	AA	CRC	
NEG	NEG	1195	8	15	0	1218
NAA	NEG	7	456	17	1	481
AA	NEG	5	13	257	2	277
CRC	NEG	1	0	0	172	173

Table 2: Error analysis for the BioBERT model in the training and validation data.

tion included “tumor,” “invasion,” and “carcinoma.” High-scoring tokens related to AA and NAA classification included “tub”, “##umour”, and “##eno.” This model interpretability analysis helped to confirm that our NLP model is able to leverage key terms that match domain knowledge.

5 Conclusion

Determining cancer status and characterizing precancerous lesions are critical and time-consuming steps for the development and evaluation of diagnostic tests for colorectal cancer screening. Here we achieved a 0.914 macro-F1 score for cancer status classification with transformer models fine-tuned using BioBERT. Informed by the domain knowledge and error analyses, we identified lesion size as a critical factor for differentiating between AAs and NAAs, but one that was not efficiently captured in BioBERT context-dependent embeddings. Using an ensemble model combining a fine-tuned BioBERT model and a lesion size named entity recognition model, we further improved classification performance to a macro-F1

score of 0.923. The lesion size extraction model was developed through transfer learning, using a transformer model trained in a non-biomedical domain. We showed that directly fine-tuning of transformer models was inadequate for domain-specific NLP tasks, and that precise feature engineering and use of ensemble models was needed to improve classification performance. Overall, we provided an accurate algorithm for characterizing precancerous cases that may help to improve early colorectal cancer detection and prevention, and a model training framework that leverages advanced NLP techniques to address complex disease phenotyping tasks in biomedical domain.

6 Limitations

One limitation of this work is that we could not fully evaluate how use of scanned reports and OCR affects performance as compared to use of electronic reports, due to a lack of dataset with paired scanned and electronic formats. The scanned format makes the selection of relevant sections from the colonoscopy and pathology reports challenging. We used fuzzy matching of selected keywords to identify sections that are likely important for diagnosis, but this process might introduce bias. Additionally, the digitization process by OCR results in errors in keywords and numerical values. For example, we observed “tubulovillous” was misrecognized as “tubulovillaus” and “0.2cm” is misrecognized as “0:2cm”. This could affect the perfor-

Model	Train			Val			Test		
	F1	precision	recall	F1	precision	recall	F1	precision	recall
FT_BERT	0.316	0.240	0.450	0.174	0.142	0.225	0.202	0.159	0.273
FT_XLM_RoBERTa	0.471	0.372	0.644	0.259	0.197	0.278	0.243	0.186	0.351
OntoNotes_XLM_RoBERTa	0.395	0.275	0.702	0.360	0.243	0.695	0.508	0.398	0.703
TL_OntoNotes_XLM_RoBERTa	0.933	0.911	0.956	0.874	0.856	0.893	0.757	0.761	0.753

Table 3: NER model performance. FT_BERT: direct fine-tuned BERT model. FT_XLM_RoBERTa: direct fine-tuned XLM_RoBERTa model. OntoNotes_XLM_RoBERTa: XLM_RoBERTa model that has been fine-tuned on OntoNotes dataset. TL_OntoNotes_XLM_RoBERTa: XLM_RoBERTa model that has been fine-tuned on OntoNotes dataset and pathology lesion size dataset.

mance in NER and final cancer status classification. Future work could include evaluating other OCR tools besides Google Vision.

Another limitation is the lack of validation studies using an external dataset. It is known that health records vary substantially in both formats and content. Studies have been done to transform pathology reports to use standardized terminologies and diagnoses (Kim et al., 2020; Ryu et al., 2020). Even though the dataset used in this study is collected from 68 collection sites across the US, the sample size is still relatively small and may not fully capture the variabilities of real-world data.

7 Acknowledgments

This work was supported by Freenome. We thank Paul Tittel and Michael Widrich for helpful discussion; Chuanbo Xu for assistance in acquiring the clinical documents; Amit Pasupathy for helping to collect the reports; and Anooj Patel and David Liu for support with the computational and machine learning infrastructure.

References

- Abdulrahman K AAIAbdulsalam, Jennifer H Garvin, Andrew Redd, Marjorie E Carter, Carol Sweeny, and Stephane M Meystre. 2018. Automated extraction and classification of cancer stage mentions from unstructured text fields in a central cancer registry. *AMIA Summits on Translational Science Proceedings*, 2018:16.
- Mohammed Alawad, Shang Gao, John X Qiu, Hong Jun Yoon, J Blair Christian, Lynne Penberthy, Brent Mumphrey, Xiao-Cheng Wu, Linda Coyle, and Georgia Tourassi. 2020. Automatic extraction of cancer registry reportable information from free-text pathology reports using multitask convolutional neural networks. *Journal of the American Medical Informatics Association*, 27(1):89–98.
- 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*.
- Shang Gao, John X Qiu, Mohammed Alawad, Jacob D Hinkle, Noah Schaefferkoetter, Hong-Jun Yoon, Blair Christian, Paul A Fearn, Lynne Penberthy, Xiao-Cheng Wu, et al. 2019. Classifying cancer pathology reports with hierarchical self-attention networks. *Artificial intelligence in medicine*, 101:101726.
- Shang Gao, Michael T Young, John X Qiu, Hong-Jun Yoon, James B Christian, Paul A Fearn, Georgia D Tourassi, and Arvind Ramanathan. 2018. Hierarchical attention networks for information extraction from cancer pathology reports. *Journal of the American Medical Informatics Association*, 25(3):321–330.
- Timothy D Imler, Justin Morea, Charles Kahi, and Thomas F Imperiale. 2013. Natural language processing accurately categorizes findings from colonoscopy and pathology reports. *Clinical Gastroenterology and Hepatology*, 11(6):689–694.
- Thomas F Imperiale, David F Ransohoff, Steven H Itzkowitz, Theodore R Levin, Philip Lavin, Graham P Lidgard, David A Ahlquist, and Barry M Berger. 2014. Multitarget stool dna testing for colorectal-cancer screening. *New England Journal of Medicine*, 370(14):1287–1297.
- Audelaure Junca, Gaëlle Tachon, Camille Evrard, Claire Villalva, Eric Frouin, Lucie Karayan-Tapon, and David Tougeron. 2020. Detection of colorectal cancer and advanced adenoma by liquid biopsy (decalib study): The ddpcr challenge. *Cancers*, 12(6):1482.
- Baek-hui Kim, Joon Mee Kim, Gyeong Hoon Kang, Hee Jin Chang, Dong Wook Kang, Jung Ho Kim, Jeong Mo Bae, An Na Seo, Ho Sung Park, Yun Kyung Kang, et al. 2020. Standardized pathology report for colorectal cancer. *Journal of pathology and translational medicine*, 54(1):1–19.
- Y Kim. 2019. Convolutional neural networks for sentence classification. arxiv 2014. *arXiv preprint arXiv:1408.5882*.
- W Ted Klein and L Havener. 2011. Standards for cancer registries volume v: Pathology laboratory electronic reporting.
- Jeffrey K Lee, Christopher D Jensen, Theodore R Levin, Ann G Zauber, Chyke A Doubeni, Wei K Zhao, and Douglas A Corley. 2019. Accurate identification of colonoscopy quality and polyp findings using natural language processing. *Journal of clinical gastroenterology*, 53(1):e25.

- 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.
- Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Ros-tamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816.
- Yue Li and David Martinez. 2010. Information extrac-tion of multiple categories from pathology reports. In *Proceedings of the Australasian Language Tech-nology Association Workshop 2010*, pages 41–48.
- David A Lieberman, Douglas K Rex, Sidney J Winawer, Francis M Giardiello, David A John-son, and Theodore R Levin. 2012. Guidelines for colonoscopy surveillance after screening and polypectomy: a consensus update by the us multi-society task force on colorectal cancer. *Gastroen-terology*, 143(3):844–857.
- David Martinez and Yue Li. 2011. Information extrac-tion from pathology reports in a hospital setting. In *Proceedings of the 20th ACM international confer-ence on Information and knowledge management*, pages 1877–1882.
- Anobel Y Odisho, Briton Park, Nicholas Altieri, John DeNero, Matthew R Cooperberg, Peter R Carroll, and Bin Yu. 2020. Natural language processing systems for pathology parsing in limited data environments with uncertainty estimation. *JAMIA open*, 3(3):431–438.
- Briton Park, Nicholas Altieri, John DeNero, Anobel Y Odisho, and Bin Yu. 2021. Improving natural lan-guage information extraction from cancer pathology reports using transfer learning and zero-shot string similarity. *JAMIA open*, 4(3):ooab085.
- Sam Preston, Mu Wei, Rajesh Rao, Robert Tinn, Naoto Usuyama, Michael Lucas, Roshanthi Weerasinghe, Soohee Lee, Brian Piening, Paul Tittel, et al. 2022. Towards structuring real-world data at scale: Deep learning for extracting key oncology information from clinical text with patient-level supervision. *arXiv preprint arXiv:2203.10442*.
- Girish Putcha, Lauren N Carroll, Tarun Chandra, and Andrew Piscitello. 2022a. Interception versus pre-vention in cancer screening in a medicare population: Results from the crc-maps model.
- Girish Putcha, Chuanbo Xu, MPH Aasma Shaukat, MD, and Theodore R Levin. 2022b. Prevention of col-orectal cancer through multiomics blood testing: The preempt crc study.
- Gottumukkala S Raju, Phillip J Lum, Rebecca S Slack, Selvi Thirumurthi, Patrick M Lynch, Ethan Miller, Brian R Weston, Marta L Davila, Manoop S Bhutani, Mehnaz A Shafi, et al. 2015. Natural language processing as an alternative to manual reporting of colonoscopy quality metrics. *Gastrointestinal en-doscopy*, 82(3):512–519.
- Gottumukkala S Raju, Vikram Vadyala, Rebecca Slack, Somashekar G Krishna, William A Ross, Patrick M Lynch, Robert S Bresalier, Ernest Hawk, and John R Stroehlein. 2013. Adenoma detection in patients undergoing a comprehensive colonoscopy screening. *Cancer medicine*, 2(3):391–402.
- Borim Ryu, Eunsil Yoon, Seok Kim, Sejoon Lee, Hyun-young Baek, Soyoung Yi, Hee Young Na, Ji-Won Kim, Rong-Min Baek, Hee Hwang, et al. 2020. Transformation of pathology reports into the com-mon data model with oncology module: use case for colon cancer. *Journal of medical Internet research*, 22(12):e18526.
- Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. 2010. Mayo clin-ical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and ap-plications. *Journal of the American Medical Infor-matics Association*, 17(5):507–513.
- Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of bayesian opti-mization. *Proceedings of the IEEE*, 104(1):148–175.
- Aasma Shaukat, Charles J Kahi, Carol A Burke, Linda Rabeneck, Bryan G Sauer, and Douglas K Rex. 2021. Acg clinical guidelines: colorectal cancer screening 2021. *Official journal of the American College of Gastroenterology | ACG*, 116(3):458–479.
- Rebecca L Siegel, Kimberly D Miller, and Ahmedin Jemal. 2019. Cancer statistics, 2019. *CA: a cancer journal for clinicians*, 69(1):7–34.
- Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.
- Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Martha Palmer, Nianwen Xue, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, et al. 2011. Ontonotes release 4.0. *LDC2011T03, Philadelphia, Penn.: Linguistic Data Consortium*.
- Sidney J Winawer and Ann G Zauber. 2002. The ad-vanced adenoma as the primary target of screening. *Gastrointestinal Endoscopy Clinics*, 12(1):1–9.
- Jialun Wu, Kaiwen Tang, Haichuan Zhang, Chunbao Wang, and Chen Li. 2020. Structured informa-tion extraction of pathology reports with attention-based graph convolutional network. In *2020 IEEE*

International Conference on Bioinformatics and Biomedicine (BIBM), pages 2395–2402. IEEE.

Adam Yala, Regina Barzilay, Laura Salama, Molly Griffin, Grace Sollender, Aditya Bardia, Constance Lehman, Juliette M Buckley, Suzanne B Coopey, Fernanda Polubriaginof, et al. 2017. Using machine learning to parse breast pathology reports. *Breast cancer research and treatment*, 161(2):203–211.

Yizhen Zhong, Luke Rasmussen, Yu Deng, Jennifer Pacheco, Maureen Smith, Justin Starren, Wei-Qi Wei, Peter Speltz, Joshua Denny, Nephi Walton, et al. 2018. Characterizing design patterns of ehr-driven phenotype extraction algorithms. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1143–1146. IEEE.

A Appendix

A.1 CRC status annotation criteria

- CRC
 - All stages (I-IV)
- Advanced adenoma (AA)
 - Adenoma with carcinoma *in situ* or high-grade dysplasia, any size
 - Adenoma, any villous features, any size
 - Adenoma ≥ 1.0 cm in size
 - Serrated lesion, ≥ 1.0 cm in size, including sessile serrated adenoma/polyp (SSA/P) with or without cytological dysplasia and hyperplastic polyps (HP) ≥ 1.0 cm
 - Traditional serrated adenoma (TSA), any size
- Non-advanced adenoma (NAA)
 - Any number of adenomas, all < 1.0 cm in size, non-advanced
- Negative (NEG)
 - All SSA/P < 1.0 cm and HP < 1.0 cm NOT in sigmoid or rectum
 - HP < 1.0 cm in the sigmoid or rectum
 - Negative upon histopathological review
 - No findings on colonoscopy, no histopathological review

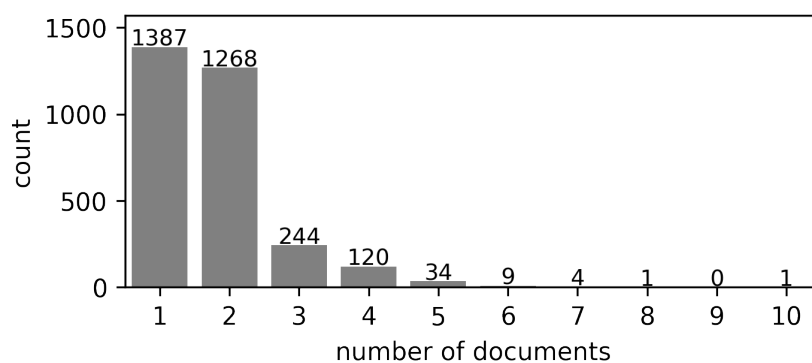


Figure A1: Distribution of document number per patient.

Sample Count	NEG	NAA	AA	CRC	Total
Train+Val set	1,221	482	273	173	2,149
Test set	515	183	146	75	919

Table A1: Sample count for combined training and validation set and for test set for cancer status classification model.

Model	Train			Validation			Test		
	F1	precision	recall	F1	precision	recall	F1	precision	recall
BoW	1.000	1.000	1.000	0.909	0.926	0.897	0.839	0.845	0.838
CNN	0.998	0.998	1.000	0.920	0.932	0.911	0.883	0.886	0.882
BioBERT	0.960	0.955	0.965	0.946	0.946	0.946	0.914	0.904	0.925
BioBERT+Lesion Size	1.000	1.000	1.000	0.921	0.921	0.927	0.923	0.920	0.930

Table A2: Model performance for cancer status classification in training, validation and test sets.

	Hyperparameters
BoW	TfidfVectorizer: <ul style="list-style-type: none"> • Minimum frequency for building vocabulary: 0~100% • Maximum number of words: 800~2500 SVM: <ul style="list-style-type: none"> • Regularization strength C: 0.001~1000 • Kernel: {linear, rbf, poly}
CNN	CNN: <ul style="list-style-type: none"> • Embedding size: {64, 96, 128, 200, 256, 512} • Size of kernels: {[3], [3, 6], [3, 4, 5], [3, 6, 9], [3, 4, 5, 6]} • Number of output channels: {32, 64, 100, 128, 256} Training: <ul style="list-style-type: none"> • Batch size: {16, 32, 50, 64, 128} • Learning rate: {0.001, 0.005, 0.01, 0.05, 0.1} • Dropout: 0.01~0.5 • Number of epochs: 256~2048 • Early stopping patience: 20~50 • Weight decay: {0.0, 0.01, 0.05, 0.1, 0.15}
BioBERT	Fine-tuning: <ul style="list-style-type: none"> • Batch size: {4, 8, 10, 12} • Learning_rate: 1.0e-7~5e-5 • Dropout: 0.001~0.5 • Number of fine-tuning epochs: {5, 6, 7, 8, 9, 10}

Table A3: Searched Hyperparameters for the cancer status classification models. We conducted 100 iterations of hyperparameter search for each of the cancer status classification models via Hyperband-enabled early-stopping (Li et al., 2017) Bayesian optimization (Snoek et al., 2012; Shahriari et al., 2015) using *Weights & Biases’ sweeps*. The best hyperparameters were determined using macro-F1 in the validation set. For NER model training, we used batch size=4, learning rate=5e-5, warmup_steps=2, lower_case=True, and n_epoch=10.

Developing Prefix-Tuning Models for Hierarchical Text Classification

Lei Chen and Houwei Chou

Rakuten Institute of Technology (RIT)

Boston, MA

{lei.a.chen,houwei.chou}@rakuten.com

Xiaodan Zhu

Ingenuity Labs Research Institute & ECE

Queen's University, Canada

xiaodan.zhu@queensu.ca

Abstract

Hierarchical text classification (HTC) is a key problem and task in many industrial applications, which aims to predict labels organized in a hierarchy for given input text. For example, HTC can group the descriptions of online products into a taxonomy or organizing customer reviews into a hierarchy of categories. In real-life applications, while Pre-trained Language Models (PLMs) have dominated many NLP tasks, they face significant challenges too—the conventional fine-tuning process needs to modify and save models with a huge number of parameters. This is becoming more critical for HTC in both global and local modelling—the latter needs to learn multiple classifiers at different levels/nodes in a hierarchy. The concern will be even more serious since PLM sizes are continuing to increase in order to attain more competitive performances. Most recently, prefix tuning has become a very attractive technology by only tuning and saving a tiny set of parameters. Exploring prefix turning for HTC is hence highly desirable and has timely impact. In this paper, we investigate prefix tuning on HTC in two typical setups: local and global HTC. Our experiment shows that the prefix-tuning model only needs less than 1% of parameters and can achieve performance comparable to regular full fine-tuning. We demonstrate that using contrastive learning in learning prefix vectors can further improve HTC performance.

1 Introduction

Hierarchical text classification (HTC) is a key task in many industrial applications. Typically, a large number of labels are defined and organized in a taxonomic tree. How to accurately and efficiently predict texts into label paths in the label hierarchies is an important capacity in high demand. For example, many e-commerce applications need to assign an online product to a path in the label hierarchy, e.g., *beverage*→*coffee*→*instant coffee* or *beverage*→*tea*→*oolong tea*. Identifying these la-

bel paths allows the information to be easily accessed by down-stream applications and human users.

In the past few years, Pre-trained Language Models (PLMs) have become a dominant solution for most natural language processing (NLP) applications. However, PLM models often contain a very large number of parameters, and the model sizes keep increasing, which can put a heavy burden on HTC applications. As an example, HTC often benefits from building a number of local models to fully utilize label hierarchies. Instead of training one model as in *global* HTC modelling, *local* HTC models rely on and leverage several inner classifiers (Peng et al., 2018). Figure 1 shows that when building a local HTC model for separating various types of drinks into the beverage category, the model sizes dramatically increase along with the increase of hierarchy levels. (More discussion about local and global HTC models can be found in Section 2).

Most recently, prefix tuning (Li and Liang, 2021; Lester et al., 2021) has become a very attractive technology by only tuning and saving a tiny set of parameters compared that of a fully fine-tuned model. Exploring prefix turning for HTC is hence highly desirable and has timely impact. In this paper, we investigate prefix tuning on HTC in two typical setups: local and global HTC. Our experiment shows that the prefix-tuning model only needs less than 1% of parameters and can achieve performance comparable to regular full fine-tuning. We demonstrate that using contrastive learning in learning prefix vectors can further improve HTC performance.

In brief, our contributions are summarized as follows:

- To the best of our knowledge, this is the first systematic study to develop prefix-tuning for HTC

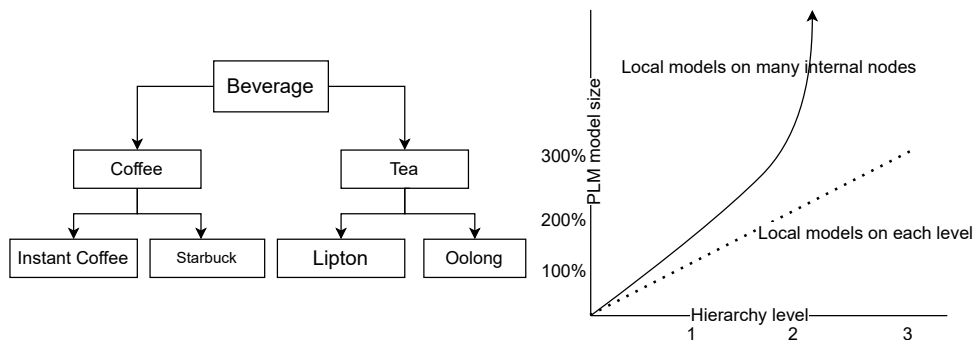


Figure 1: An illustration to show that local HTC model will face a size issue when using PLM models to be classifiers.

- Following local HTC modelling, we examine different architectures to leverage prefix vectors learned at different levels of label hierarchies and provide results about our best practice.
- In the global HTC strategy, we propose to add a self-training step built on a contrastive learning (CL) loss and this shows to improve performance.
- We provide detailed results on two HTC datasets and the analyses to show how the models work.

2 Related work

There are two major means of handling label hierarchies for HTC, i.e., the local and global approach (Zhou et al., 2020). The local approach builds a number of classifiers on different label levels or on many internal nodes in a label hierarchy but the global approach develops a single classifier to predict all labels that are flattened from the label hierarchy.

Shimura et al. (2018) developed convolution neural network (CNN) based local models at each level of label hierarchies and proposed to use the trained CNN at the higher level to initialize the CNN at a lower level. This *transferring* approach that considers inter-connections among the CNN models in a hierarchy showed to improve HTC performance. Regarding the global HTC, a straightforward method is flattening labels’ hierarchical structure into a flat list and modelling the HTC simply to a multi-label classification task. Recently, a trending method is utilizing a structure encoder to retain the label hierarchy to better utilize mutual information among labels. (Zhou et al., 2020)

used a structure encoder, either a tree LSTM or a graph convolution network (GCN), to consider labels’ prior hierarchy information when learning label representations. PLMs have become a foundational paradigm on building various NLU tasks. For example, BERT (Devlin et al., 2019) has been applied to tackle the HTC task (Chen et al., 2021; Wang et al., 2022).

In parallel, contrastive learning (CL) has been found to be effective in providing high-quality encoders in a simple self-learning way. For example, in computer vision, SimCLR (Chen et al., 2020) uses the consistence between an anchor image and its transformed version and the in-consistence between the anchor and other instances in a batch (in-batch negative instances) to guide encoder training. Inspired by the success of SimCLR in computer vision, CL-based textual representation learning has become a hot research topic in NLP. SimCSE (Gao et al., 2021) uses dropout operations existing in Transformer (Vaswani et al., 2017) to provide self-augmentation and can learn effective text representations.

The CL training has been applied on the HTC task. (Chen et al., 2021) embeds both text inputs and labels (in a hierarchy) into an unified semantics space and solving the HTC via *vector matching*. When training the text encoder, a CL setup is used and it considers label hierarchy information when forming contrastive pairs. (Wang et al., 2022) uses a CL setup to train a high-quality text encoder. Label hierarchy is firstly encoded by a Graphormer (Ying et al., 2021) and the encoded label information is used to generate text variations for providing positive pairs in the CL.

Although fine-tuning PLM models enable many down-stream natural language understanding (NLU) tasks to achieve high performance, this

paradigm faces a challenge in real deployment compared to other light weight models, e.g., CNN. Also, PLMs contain a large number of parameters and the model sizes have been exploding in recent years for reaching more competitive performance and the trend is continuing. When deploying the fine-tuned PLM models, all model parameters (updated in the fine-tuning process) need be stored. When many such PLM models need be stored, for example, for local HTC modelling, the required models sizes can be very large. To use PLMs in a more space-efficient way, previous efforts, such as fine-tuning only several top layers in a PLM or fine-tuning an adapter, are proposed (He et al., 2022). Unlike them, prefix-tuning (Li and Liang, 2021; Lester et al., 2021) only learns prefix vectors to trigger a PLM, which is frozen and cannot be tuned, to output the text representations fitting to the targeted domain better. In addition, (Liu et al., 2022) extended the prefix-tuning on NLU tasks by using prefix vectors on each PLM layer and dropped several components in a conventional prompt-tuning, e.g., verbalizer.

3 Exploring Effective Prefix Tuning for Hierarchical Classification

Let x denote the text input, Y a label hierarchy and y a specific category label path in Y . HTC aims to solve a multi-label categorization task: given textual input x , HTC learns to predict possible label paths y in the hierarchy Y . As discussed above, when developing HTC in industrial applications, PLM-based models face model-size issues, which is becoming more serious as PLM sizes are continuing to increase. To tackle the challenge, we investigate soft prefix prompt (SPP) tuning on HTC. We propose to explore the models in two typical approaches. In Section 3.1 below, we explore a transferring approach to better train SPP vectors among different label levels in the local HTC modelling. Section 3.2 explores global HTC models in which we propose to add a CL-based self-training step.

3.1 SPP tuning considering hierarchical information

Figure 2 depicts two approaches of fine-tuning a PLM model for text classification. The left subfigure shows the conventional [CLS]-tuning, in which the [CLS] token is appended in front of the input text x . The entire text sequence goes through mul-

iple Transformer layers in a PLM model. Built on that, the hidden output $h_{[CLS]}$ at the final layer serves as the representation for x . The $h_{[CLS]}$ passes through a linear classifier layer (denoted as CLF in Figure 2) to make predictions. Using the fine-tuning data, losses can be fed back into the model and all parameters in both the classifier head and the PLM model are accordingly tuned. Unlike that, the right subfigure highlights the process of (SPP) tuning (Liu et al., 2022), in which the entire PLM model is frozen and will not be updated during fine-tuning. For the embedding layer and each of PLM layers, tunable SPP vectors, which have a much smaller sizes compared to the PLM, are tuned to trigger the frozen PLM to output a more informative $h_{[CLS]}$ for prediction.

When using the SPP-tuning, the HTC model focuses on a set of SPP vectors. In the local HTC model, these SPP vectors on different locations/layers in a hierarchy may have some inter-constraints and therefore training them by considering their topological relationship in the hierarchy is our first consideration.

Specifically, Figure 3 depicts how we perform SPP-tuning on adjacent hierarchy layers. Subfigure (a) shows a basic solution in which SPP vectors on different levels of the hierarchy are trained independently without considering any inter-level connections. However, subfigure (b) shows that the trained SPP vector at a higher level is used to initialize a part of SPP vector at a lower level. The motivation is that knowledge learned in the upper layer prefix can help inform the low layer decision. In our study, we propose to assign lower-level SPP vectors longer than the SPP vectors at a higher level since the former needs handle more labels. In addition, we propose and investigate the architecture in subfigure (c) where the SPP vector at the higher level is transformed to a longer vector to initialize the lower-level SPP vector by using a fully-connected neural network.

3.2 Global model using contrastive learning when doing SPP-tuning

The other typical setup is investigating global HTC models. As shown in the right subfigure of Figure 2, when training SPP vectors, the loss after the classifier layer is used in supervised learning. Unlike the local modelling, here we do not transfer prefix among different layers. When develop the model, inspired by the success of using self-learning to

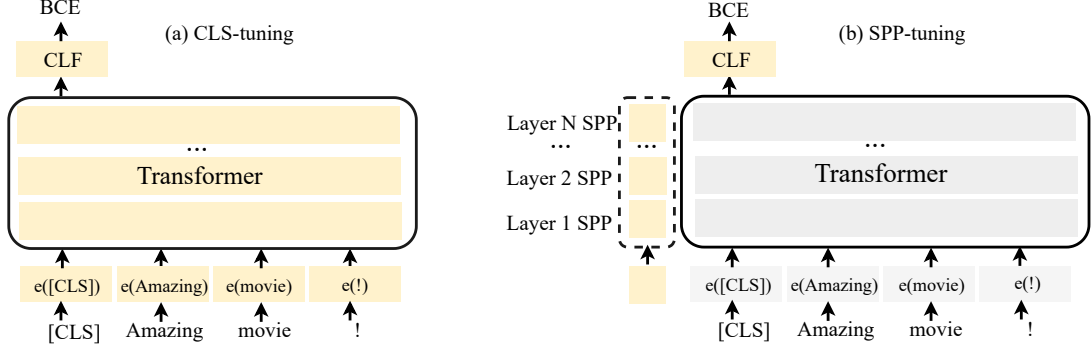


Figure 2: (a) shows conventional [CLS]-tuning for using PLM models. Note that all of parameters in a PLM need tuning and are shown in a light yellow color. In a contrast, (b) shows Soft Prefix Prompt (SPP) tuning, in which a frozen PLM model is used and only small-sized SPP vectors (on embedding input and each PLM layer) are tuned.

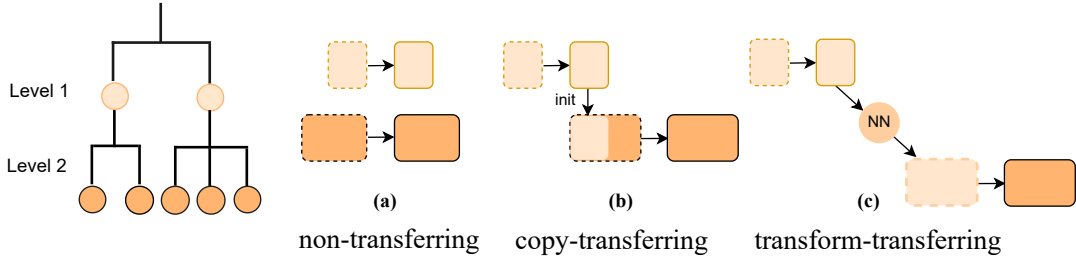


Figure 3: When solving HTC using a local model strategy, there are several ways to train prefix vectors for local models on each label level. (a) shows that vectors from upper and lower levels can be trained separately, (b) shows that lower level vectors can be initialized based on trained vectors at an upper level, and (c) shows that lower level vectors can be initialized based on the trained vectors at an upper level and go through a neural network (NN) transformation. Rectangles at the two levels refer to SPP vectors, from being initialized (enclosed in dash lines) to being fine-tuned (enclosed in solid lines).

learn proper representations, contrastive learning is found to be beneficial when being applied with SPP in the global modelling.

Specifically in our SPP-tuning setup, we follow the SimCSE (Gao et al., 2021) contrastive approach to feed inputs into a PLM model twice to obtain a data anchor and its positive pair.

For a text title \mathbf{x} , we append the SPP vector (V_{spp}) to [CLS]. Then we obtain a text representation \mathbf{t} with a BERT encoder $BERT(*, d)$ where d is a dropout mask, and a projection function g , which uses a simple multiple layer perception (MLP) structure.

$$\mathbf{t} = g(BERT(V_{spp} : [CLS] : \mathbf{x}, d)) \quad (1)$$

To obtain a positive pair, SimCSE runs the same text title throughout the Transformer encoder pipeline with a different dropout mask d^+ .

$$\mathbf{t}^+ = g(BERT(V_{spp} : [CLS] : \mathbf{x}, d^+)) \quad (2)$$

For the i^{th} text title, the training objective of

SimCSE is as follows:

$$\mathcal{L}_i = -\log \frac{\exp(\text{sim}(\mathbf{t}_i, \mathbf{t}_i^+)/\tau)}{\sum_{j=1}^{N, j \neq i} \exp(\text{sim}(\mathbf{t}_i, \mathbf{t}_j)/\tau)} \quad (3)$$

For a mini-batch of N text titles, where $\text{sim}(*, *)$ represents a similarity computation and τ is the temperature. The total loss computed by SimCSE, \mathcal{L}_{simCSE} , is an average among all text titles in the mini-batch, $\sum_{i=1}^N \mathcal{L}_i / N$. By running an optimization to keep reducing \mathcal{L}_{simCSE} , the SPP vectors on multiple layers of the BERT model can be tuned prior to applying the supervised fine-tuning. To the best of our knowledge, this is the first work to apply CL pre-training on an NLU task in SPP-tuning.

4 Experiments

4.1 Datasets and Evaluation

We perform our study on the widely used Web of Science (WoS) (Kowsari et al., 2017) and Amazon review dataset (McAuley et al., 2015; He and

Dataset	levels	Train size	Dev size	Test size	classes
WoS	2	33,070	7,518	9,397	141
Amazon Beauty	5	116,240	29,061	62,273	241

Table 1: Our experiments use both academic benchmark data set, WoS, which has been widely used in previous HTC research (Chen et al., 2021; Wang et al., 2022), and also an industry data from Amazon review dataset.

McAuley, 2016), focusing on the Beauty category for comparison and analysis. WoS contains abstracts of published papers from Web of Science and Amazon review contains titles of online products. For each instance in WoS, there is only a single label path. However, for each instance in Amazon Beauty, there could be more than one possible label paths. Regarding these two datasets, more statistic details are reported in Table 1. Similar to previous works, we measure the experimental results by using micro-F1 (denoted as mi-F1) and macro-F1 (denoted as ma-F1) to value performances per instance and per label respectively.

4.2 HTC models

We consider a variety of HTC models:

- **CLS-tuning**: A global model using a binary cross entropy (BCE) loss to train a HTC model as a multi-label text classifier.
- **Local SPP-tuning**: A local model consisting of two multi-label text classifiers trained with SPP. Between SPP vectors at the top and bottom label levels, there are three ways to train: (a) **non-transferring** refers to the basic strategy described in Section 3.1, training two sets of SPP vectors independently, (b) **copy-transferring** refers to using the trained SPP vector at the top level to initialize the corresponding portion in the longer SPP vector at the bottom level, and (c) **transform-transferring** refers to using a neural network to transform the SPP vector at the upper level to longer vector to initialize the SPP vector at lower level. Since the label hierarchy in the WoS data contains exactly two levels, we built local models on each level.
- **Global SPP-tuning**: a global model trained by using the SPP-tuning, and the BCE loss is used to train the SPP vectors
- **Global SPP-tuning with CL**: before training SPP vectors by the BCE loss, as described in

Section 3.2, a contrastive learning (CL) self-learning step is used to better initialize SPP vectors.

For the PLM model, we used BERT-base provided in the Hugging Face’s Transformer library (Wolf et al., 2020). The batch size is set to be 48. The optimizer is Adam with a learning rate of $1e^{-2}$ for the SPP-tuning and the learning rate of $2e^{-5}$ for CLS-tuning¹. We implemented all above-mentioned models based on the source code² provided by (Liu et al., 2022) in PyTorch. We trained these models in an end-to-end way on the training set for up to 40 epochs and use an early stop if there was no any performance gain for consecutive 5 epochs on the development set. SPP vector length is an important hyper-parameter controlling SPP-tuning performance. Typically for simple NLU tasks, short SPP vectors, e.g., shorter than 20, could work sufficiently. Hence, we did a grid search among SPP vector lengths from 5 to 40 and found optimal SPP vector lengths for local and global models respectively. When conducting CL pre-training, we set the batch size to be 64 to maintain enough in-batch negative samples and used a temperature (τ) of 0.1. We conducted the CL pre-training for 10 epochs.

4.3 Results

Table 2 reports the result of comparing the three training strategies for building HTC models built on SPP-tuning. When using SPP-tuning to train a global model on the WoS data, we can find that by only using 0.46% of model parameters as used in CLS-tuning, we can achieve a performance even higher than that obtained by using the CLS-tuning. Among the three local HTC models based on SPP-tuning, the non-transferring approach yields the lowest performance, even worse than the result of using CSL-tuning. The transform-transferring approach works better than non-transferring and

¹(Liu et al., 2022) uses higher learning rate than what is used by CLS-tuning. We found this choice makes our SSP-tuning work.

²<https://github.com/THUDM/P-tuning-v2>

Model	Detail	Mi-F1	Ma-F1	Parameters(%)
CLS-tuning	global, BCE	85.89	79.22	100%
SPP-tuning	global, BCE, $ V_{spp} = 30$	86.84	79.77	0.46%
SPP-tuning	local, non-transferring, $ V_{spp}^{top} = 10$ $ V_{spp}^{bottom} = 20$	86.84	79.12	0.46%
SPP-tuning	local, transform-transferring, $ V_{spp}^{top} = 10$ $ V_{spp}^{bottom} = 20$	86.93	79.33	0.46%
SPP-tuning	local, copy-transferring, $ V_{spp}^{top} = 10$ $ V_{spp}^{bottom} = 20$	87.24	79.98	0.46%
SPP-tuning	local, copy-transferring, $ V_{spp}^{top} = 10$ $ V_{spp}^{bottom} = 30$	87.34	80.09	0.66%

Table 2: HTC models on WoS dataset. By using a BERT-base, various fine-tuning methods, i.e., CLS-tuning, global model using SPP-tuning, and local models using SPP-tuning, are compared.

Model	Detail	WoS			Amazon Beauty		
		Mi-F1	Ma-F1	Parameters (%)	Mi-F1	Ma-F1	Parameters (%)
CLS-tuning	BCE	85.89	79.22	100.00%	87.49	63.38	100.00%
SPP-tuning	BCE	86.84	79.77	0.46%	88.00	61.36	0.50%
SPP-tuning	CL→BCE	87.55	80.07	0.46%	88.14	62.07	0.50%

Table 3: Global HTC models on both WoS and Amazon datasets. When training by SPP-tuning, we proposed adding a CL pre-training stage and this turns out to improve HTC performance.

the best performance is from copy-transferring approach. When keep increasing V_{spp}^{bottom} from 20 to 30, the performance can be improved further. It shows that SPP vectors trained at a higher level need be used intact when initializing SPP vectors at lower levels. Also, both transferring approaches work better than non-transferring.

Table 3 reports the results of comparing the two types of training losses when training a global model based on SPP-tuning. We can see that on the two data sets, WoS and Amazon Beauty, the SPP-tuning only using the BCE loss is worse than the proposed model that leverages CL pre-training.

Note that when only using 0.5% of the parameters used in CLS-tuning, on the Amazon Beauty data with 241 labels, the SPP-tuning model has a performance comparable to CLS-tuning. A slight gain is actually observed on micro-F1, although macro-F1 has some drop from 63.38% to 61.36%. We show that after adding CL self-training prior to fine-tuning SPP vectors, macro-F1 is still lower than what we can obtain when using CLS-tuning. This is worth more investigations to evaluate SPP-tuning approach comprehensively, on labels with both sufficient and sparse training instances.

5 Conclusions and Future Work

HTC is a key task in many industrial applications. The conventional fine-tuning process needs to modify and save models that have a large number of parameters. This has become a more significant

concern as PLM sizes are continuing to increase in the foreseeable future. In this paper, we investigate prefix tuning on HTC in two typical setups: local and global HTC. To the best of our knowledge, this is the first systematic study towards developing prefix-tuning for HTC in these typical architectures.

In local HTC modelling, we examine different architectures to leverage prefix vectors learned at different levels of label hierarchies and provide results about our best practice. We found that SPP vectors trained at a higher level can be utilized to initialize a portion of SPP vectors at a lower level of the hierarchy and such a vector transferring strategy is beneficial. For SPP vectors, using them intact works better than using their transformed version. In the global HTC strategy, we propose to add a self-training step built on a contrastive learning (CL) loss. On both WoS and Amazon datasets, such a CL pre-training is found to be helpful on improving model performance.

For future work, we will extend the current work to study long-tailed labels which is very common in many applications. Also, how to use labels' hierarchical information that can be represented by structural encoders is worth studying in SPP-tuning.

References

- Haibin Chen, Qianli Ma, Zhenxi Lin, and Jianguye Yan. 2021. [Hierarchy-aware Label Semantics Matching Network for Hierarchical Text Classification](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4370–4379, Online. Association for Computational Linguistics.
- 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](#). *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. [SimCSE: Simple Contrastive Learning of Sentence Embeddings](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. [Towards a Unified View of Parameter-Efficient Transfer Learning](#).
- Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517.
- Kamran Kowsari, Donald E. Brown, Mojtaba Heidarysafa, Kiana Jafari Meimandi, Matthew S. Gerber, and Laura E. Barnes. 2017. [HDLTex: Hierarchical deep learning for text classification](#). In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. [The Power of Scale for Parameter-Efficient Prompt Tuning](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Xiang Lisa Li and Percy Liang. 2021. [Prefix-Tuning: Optimizing Continuous Prompts for Generation](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.
- Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. [P-Tuning: Prompt Tuning Can Be Comparable to Fine-tuning Across Scales and Tasks](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.
- Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52.
- Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pages 1063–1072.
- Kazuya Shimura, Jiyi Li, and Fumiyo Fukumoto. 2018. [HFT-CNN: Learning Hierarchical Category Structure for Multi-label Short Text Categorization](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 811–816, Brussels, Belgium. 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. *Advances in neural information processing systems*, 30:5998–6008.
- Zihan Wang, Peiyi Wang, Lianzhe Huang, Xin Sun, and Houfeng Wang. 2022. [Incorporating Hierarchy into Text Encoder: a Contrastive Learning Approach for Hierarchical Text Classification](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7109–7119, Dublin, Ireland. 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.
- Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? *Advances in Neural Information Processing Systems*, 34:28877–28888.

Jie Zhou, Chunping Ma, Dingkun Long, Guangwei Xu, Ning Ding, Haoyu Zhang, Pengjun Xie, and Gongshen Liu. 2020. Hierarchy-aware global model for hierarchical text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1106–1117.

PAIGE: Personalized Adaptive Interactions Graph Encoder for Query Rewriting in Dialogue Systems

Daniel Biś

Amazon Alexa AI

bisdb@amazon.com

Saurabh Gupta¹

LinkedIn

saurabh3949@gmail.com

Jie Hao

Amazon Alexa AI

jieha@amazon.com

Xing Fan

Amazon Alexa AI

fanxing@amazon.com

Chenlei Guo

Amazon Alexa AI

guochenl@amazon.com

Abstract

Unexpected responses or repeated clarification questions from conversational agents detract from the users’ experience with technology meant to streamline their daily tasks. To reduce these frictions, Query Rewriting (QR) techniques replace transcripts of faulty queries with alternatives that lead to responses that satisfy the users’ needs. Despite their successes, existing QR approaches are limited in their ability to fix queries that require considering users’ personal preferences. We improve QR by proposing **Personalized Adaptive Interactions Graph Encoder (PAIGE)**. PAIGE is the first QR architecture that jointly models user’s affinities and query semantics end-to-end. The core idea is to represent previous user-agent interactions and world knowledge in a structured form — a heterogeneous graph — and apply message passing to propagate latent representations of users’ affinities to refine utterance embeddings. Using these embeddings, PAIGE can potentially provide different rewrites given the same query for users with different preferences. Our model, trained without any human-annotated data, improves the rewrite retrieval precision of state-of-the-art baselines by 12.5–17.5% while having nearly ten times fewer parameters.

1 Introduction

Facilitating seamless human-computer interactions is a fundamental goal of conversational AI agents such as Alexa, Cortana, and Siri. However, some user interactions lead to *frictions*, where the AI agent delivers an unexpected response or repeatedly asks the user to clarify the query. Such frictions stem from system errors such as Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU). Some aspects of the frictions are highly personalized, depending on characteristics such as the user’s demographics and interests. For example, when asking a conversational agent

to “Put on Skyfall,” one user may expect the system to play a song named “Skyfall” while another may wish to see a movie with the same title.

Query Rewriting (QR; Grbovic et al., 2015; Ponnusamy et al., 2020) aims to reduce frictions by replacing the transcripts of faulty queries with alternatives that lead to desired responses. Personalized QR systems were proposed in (Fan et al., 2021; Cho et al., 2021), which restricted rewrite candidates to the particular user’s historical requests. Such systems, discussed in Section 6, typically trained a text encoder to measure the similarity between request and rewrite. While effective, they overlook correlations between requests within a user’s dialogue history and the inter-dependencies spanning across other users’ interactions. This information can help reformulate defective and ambiguous requests when augmented with external knowledge.

To address the aforementioned limitations, we introduce a QR architecture named **Personalized Adaptive Interactions Graph Encoder (PAIGE)** that jointly models query semantics, world knowledge, and users’ preferences in an end-to-end fashion. The core idea is to represent users’ previous interactions in a heterogeneous graph that we can augment with external world knowledge (§3). The graph representation learning with Graph Neural Networks (GNNs) allows us to propagate the representations of users’ historical interactions to refine the utterance embeddings in an end-to-end manner through joint training.

To construct the heterogeneous graph, we decompose the requests into smaller semantic units such as domains, intents, utterances, entities, and NLU-hypotheses (§3). We also create nodes representing the users and link every user node to nodes representing entities (e.g., songs, artists) appearing in the user’s historical requests. Inspired by work in recommendation systems, we use cross-user connections to capture communicative intents among users (Goldberg et al., 1992; Wang et al., 2019b)

¹Work done while at Amazon Alexa AI.

and further ground the entity nodes in a knowledge graph (e.g., Wang et al., 2019b, 2020), allowing PAIGE to learn from the emerging high-order connectivities.

We cast query rewriting as a link prediction problem between an utterance node and nodes corresponding to NLU-hypotheses, that abstract away syntactic variations. Once the link to NLU-hypothesis node is predicted, we can follow the graph’s edges to select the most frequent non-defective utterance mapping to that NLU-hypothesis as our rewrite.

PAIGE is scalable in training, without the need to load the full graph in memory, and efficient at inference, without the need to re-process the entire graph. Our inductive node encoding scheme permits dynamically updating the graph to new knowledge and user interests without model re-training. We demonstrate the efficacy of our system with a detailed analysis of experiments on real-world conversation data (§5);

Our contributions are summarized as follows:

- We introduce PAIGE—a novel graph-based architecture for the task of personalized query rewriting in dialogue systems.
- We present a scalable and inductive method for joint learning of query semantics and structured user preferences in an end-to-end fashion.
- We show that modeling the high-order relations in the graph facilitates collaborative learning from customers’ collective behaviors.
- PAIGE outperforms state-of-the-art baselines (i.e., 43.8% P@1 increase) while having nearly $10\times$ fewer parameters.

2 Preliminaries

Spoken dialogue systems consist of many sequential components. When a user interacts with their device, the agent’s ASR takes the audio signal as input and transcribes it into textual utterance (*query*). Next, the transcript enters the NLU module that interprets it so that the downstream modules can satisfy the user’s request. An NLU component typically consists of domain and intent classification and entity linking, executed sequentially. As a pre-processing step for later modules in the dialogue system, the NLU module is instrumental to the system’s overall quality. One of the challenges in the NLU module is handling ambiguity or errors cascading from the previous components. *Query*

Rewriting (QR) component tackles this issue by replacing the ASR transcript with an alternative that leads to a satisfactory response for the user. Once the NLU pipeline receives a rewrite, regular data flow resumes.

2.1 Interactions Data Selection

As hand-annotating a large set of query-rewrite pairs is expensive, we use weakly-labeled data during training. Inspired by Fan et al. (2021) and Cho et al. (2021), we leverage users’ feedback to collect the datasets. For example, if a user barged in or stopped the agent’s response, the turn is defective. The details are available in Appendix A.

3 Graph Construction

The first step is building a heterogeneous graph from user-agent interactions expressed as text and semi-structured metadata. The graph will provide the computational architecture for the message passing algorithm.

Heterogeneous Graph (HG; Sun and Han (2013)). HG is defined as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node type mapping function $\tau: \mathcal{V} \rightarrow \mathcal{A}$ and a link type mapping function $\psi: \mathcal{E} \rightarrow \mathcal{R}$, where each node $v \in \mathcal{V}$ belongs to one particular node type $\tau(v) \in \mathcal{A}$, each link $\varepsilon \in \mathcal{E}$ belongs to a particular relation $\psi(\varepsilon) \in \mathcal{R}$, $|\mathcal{A}| + |\mathcal{R}| > 2$, and if two links belong to the same relation type, the two links have the same starting node types and ending node types.

3.1 Design Motivation

A simple way to build an interactions graph would be to link users with their utterances and the defective utterances with their rewrites. Unfortunately, such an approach produces a sparse graph due to the high degree of linguistic variance in the utterances and fails to capture users’ entity and domain level preferences. However, GNNs require sufficient connectivity to be effective because their efficacy stems from feature propagation and smoothing across the graph’s edges (Zhang et al., 2021).

NLU-Hypothesis. To abstract away syntactic variance in users’ requests, we group queries with similar meaning by parsing them into structured representations called NLU-hypotheses using the agent’s NLU module. Each hypothesis takes the form of “domain | intent | slot_type:slot_value.” The domain is the general topic of a query, e.g., “Weather.” The intent reflects the action the user wants to take, e.g., “PlayMusic.” Finally, the slot types/values

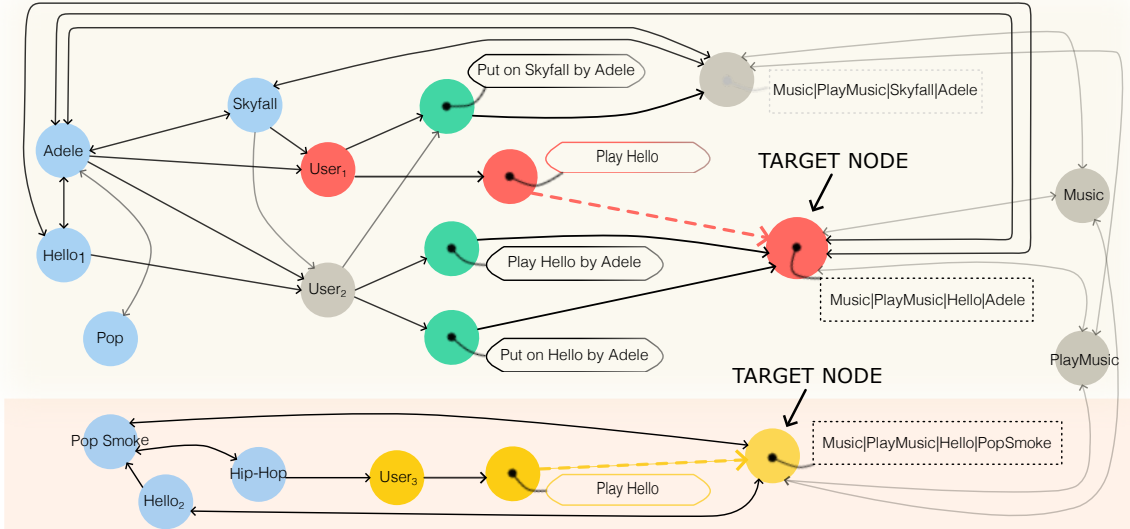


Figure 1: Customer interactions graph augmented with external knowledge. Knowledge enhances collaborative learning across users to enable reasoning-powered affinity/preference prediction. The goal is to map the two occurrences of an ambiguous utterance, *Play Hello*, to their respective interpretations (NLU-Hypotheses) for $user_1$ (red nodes) and $user_3$ (yellow nodes). To achieve this, PAIGE utilizes message passing over paths such as (Hello, Adele, $user_1$, *Play Hello*), and (Hello, Pop Smoke, $user_3$, *Play Hello*) for the two users. Nodes for non-defective queries are shared among users, e.g., *Play Skyfall by Adele* (top, green).

are results of entity labeling from the NLU module. To illustrate, the queries “*Play Hello by Adele*” and “*Put on Hello by Adele*” map to the same hypothesis: “Music | PlayMusic | SongName:Hello | ArtistName:Adele.” We use the hypotheses’ fields as “semantic units” and assign them nodes to induce a dense graph with a rich set of relations.

3.2 Graph Schema

Every distinct hypothesis, $h \in \mathcal{H}$, is assigned a node in the graph. Moreover, we create a node for each unique domain, intent, and entity (*slot*), and link them to the nodes representing the hypotheses in which they occur. Additionally, as illustrated in Figure 1, edges in our graph connect users, \mathcal{U} , to their respective utterances, \mathcal{T} , and the utterances to their corresponding hypotheses, \mathcal{H} . The NLU-hypothesis nodes act as sub-graph pooling nodes and represent groups of equivalent queries and their side information, whereas the utterance nodes represent the individual queries. The utterance nodes do not need to be stored after training; instead, they can be created on the fly to keep the adjacency matrix up to date since PAIGE uses inductive encoders for nodes with textual input features (§4.1).

There are two types of utterance nodes in our graph: non-defective and defective. Including defective queries in the graph allows to explicitly model users’ rephrase behaviors. We use historical

query-rewrite pairs to replace the hypotheses of defective queries with the ones generated for their respective rewrites. In general, utterances with different NLU hypotheses map to different nodes, even if they have the same text, e.g., the two “*Play Hello*” nodes in Figure 1. For *non-defective* queries, $c \in \mathcal{C}$, with identical text and NLU-hypothesis, we create a single utterance node to represent their text for all users, e.g., a single “*Play Skyfall by Adele*” node in Figure 1 is shared by two users. For a *defective* utterance, $b \in \mathcal{B}$, we create a distinct *defect* node for each user for whom the utterance caused friction. Consequently, each defect node has a single incoming edge from the author’s user node, and only the information relevant to the defective query’s author directly affects the embedding of that query. At inference, we create a new defect node for an utterance that is not found in the user’s dialogue history. Our task is to predict links between the new utterance nodes and the nodes associated with their NLU-hypotheses. Once the NLU-hypothesis node is predicted, we can simply follow the graph’s edges to select the most frequent non-defective utterance that maps to that NLU-hypothesis as our rewrite.

Factual Knowledge. We align the entity nodes in our graph with nodes in a knowledge graph (KG). A KG is an instance of a heterogeneous graph that consists of real-world entities and their relation-

ships. A KG is organized into (v_i, r, v_j) -triples, where $v_i, v_j \in \mathcal{V}$ are the entities, and $r \in \mathcal{R}$ is the relation type, *e.g.*, (*Adele*, *AUTHOR OF*, *Hello*).

Grounding the model in an explicit representation of knowledge facilitates rewrites that require understanding relationships that are not obvious from the user’s dialogue history alone, *i.e.*, users’ implicit preferences. We link the nodes corresponding to named entities found in each user’s queries with the corresponding user nodes. As a result, the information from KG propagates through the user nodes to the utterance nodes. Crucially, as described above, entities are also connected to the NLU-hypotheses. Since GNN acts as neighborhood smoothing, our model favors the NLU-hypotheses with neighborhoods that contain entities relevant to the user who submits the query.

4 PAIGE Model

Computing node representations involves two steps: using specialized encoders to generate nodes’ input features and applying message-passing layers to enable the features to interact and coalesce. In the message-passing step, we use relation-specific convolutional modules that aggregate feature vectors of the neighboring nodes. These modules learn to aggregate information from the node’s immediate neighborhood, and stacking K such operations effectively convolves features across the K -th order neighborhood, *i.e.*, representations of nodes depend on all the nodes that are at most K edges away.

4.1 Input Features

PAIGE uses dedicated encoders for different types of nodes to produce input embeddings for the GNN. Our inductive feature encoding design permits updates to the graph’s structure without expensive model retraining, *i.e.*, adding nodes for users, entities, and utterances. Thus, PAIGE can adapt to evolving user interests and world knowledge.

A major scalability challenge for end-to-end training of our model lies in encoding textual inputs. The reason is that the number of utterances needed to produce embeddings for a graph’s nodes grows exponentially with the number of GNN layers. Therefore, we encode textual inputs, $t_i \in \mathcal{T}$, for the utterance nodes, $\tau(v_i) \in \mathcal{T}$, using a lightweight, two-layer Bidirectional Gated Recurrent Unit (BiGRU) network.

The domain and intent nodes are the only node types for which we use a fixed vocabulary; this

is feasible because both sets change infrequently and amount to fewer than 10K vectors. While previous works tend to use fixed-size embedding tables for users or entities (Wang et al., 2019b), such an approach prohibits dynamic updates to the graph’s structure (*e.g.*, adding new users). Instead, we embed historical queries using a pre-trained RoBERTa-base model (Liu et al., 2019), and represent: *i)* *users* with the mean of embeddings of their previous queries; *ii)* *entities* with the mean of embeddings of queries in which they appear. The parameters of RoBERTa are fixed during training to prevent temporary trends in training data from leaking into initial entity representations.

Formally, a feature encoder enc_{τ_θ} embeds a node $v \in \mathcal{V}$ with type $\tau(v) \in \mathcal{A}$ as $\mathbf{x}_\tau = \text{enc}_{(\tau)_\theta}(v)$, where $\mathbf{x}_\tau \in \mathbb{R}^{d_\tau}$ is a dense feature vector. As the nodes come from different distributions, each feature encoder contains a fully connected feed-forward network that is applied to each node separately and identically to project vectors to a shared embedding space before the GNN layers.

$$\mathbf{h}_\tau^{(0)} = \phi\left(\mathbf{x}_\tau \mathbf{W}_\tau^{(1)} + \mathbf{b}_\tau^{(1)}\right) \mathbf{W}_\tau^{(2)} + \mathbf{b}_\tau^{(2)}, \quad (1)$$

where $\mathbf{h}_\tau^{(0)} \in \mathbb{R}^{d_{gnn}}$ is a node embedding, and $\mathbf{W}_\tau^{(1)} \in \mathbb{R}^{d_\tau \times d_{ffn}}$, $\mathbf{W}_\tau^{(2)} \in \mathbb{R}^{d_{ffn} \times d_{gnn}}$, $\mathbf{b}_\tau^{(1)} \in \mathbb{R}^{d_{ffn}}$, $\mathbf{b}_\tau^{(2)} \in \mathbb{R}^{d_{gnn}}$ are learnable parameters, and ϕ is a GELU activation (Hendrycks and Gimpel, 2016). We train feature encoders, except for RoBERTa, jointly with the graph neural network to enable each module to learn from other modalities.

4.2 Graph Encoder

In each layer, PAIGE propagates latent node feature information across edges of the graph while taking into account the type of an edge (Schlichtkrull et al., 2018). A single message-passing layer takes the following form

$$\bar{\mathbf{h}}_i^{(k+1)} = \phi\left(\sum_r \sum_{j \in N_r^i} \frac{\mathbf{W}_r \mathbf{h}_j^{(k)}}{c_{ijr}} + \frac{\mathbf{h}_i^{(k)}}{c_{i_r}}\right), \quad (2)$$

where $\bar{\mathbf{h}}_i^{(k)} \in \mathbb{R}^{d_{gnn}}$ is the hidden state of node v_i in the k -th layer of the neural network, r is a relation type, $\mathbf{W}_r \in \mathbb{R}^{d_{gnn} \times d_{gnn}}$ is a relation-type specific parameter matrix, ϕ is a Leaky-ReLU activation, and c_{i_r} and c_{ijr} are normalization constants.

To avoid over-smoothing, we apply residual con-

nections around each GNN layer,

$$\mathbf{h}_i^{(k+1)} = \alpha^{(k)} \text{GNN}(\mathbf{h}_i^{(k)}) + (1 - \alpha^{(k)}) \mathbf{h}_i^{(k)}, \quad (3)$$

where $\alpha^{(k)}$ is a learnable scalar parameter.

Finally, we concatenate the representations of utterance nodes, $t \in \mathcal{T}$, from the BiGRU and GNN and pass the result through a feedforward network,

$$\mathbf{h}_{v_i \in \mathcal{T}}^{\text{out}} = \phi([\text{BiGRU}(v_i) \parallel \text{GNN}(v_i)] \mathbf{W} + \mathbf{b}) \quad (4)$$

where $[\cdot \parallel \cdot]$ is concatenation, $\mathbf{W} \in \mathbb{R}^{(2 \cdot d_{\text{gnn}}) \times d_{\text{gnn}}}$ and $\mathbf{b} \in \mathbb{R}^{d_{\text{gnn}}}$ are parameters, and ϕ is a Leaky-ReLU activation. Our experiments show that concatenating the input and output embeddings of the GNN improves the QR performance by up to 5.5%.

4.3 Graph Decoder

Once the encoder maps each node $v_i \in \mathcal{V}$ to an embedding, \mathbf{h}_i , the goal of the decoder is to use these embeddings to predict labeled links in the graph. In particular, the decoder scores a (v_i, r, v_j) -triplet using a function g to represent how likely it is that the hypothesis associated with v_j is the right interpretation of the utterance associated with v_i .

$$g(v_i, r, v_j) = \sum_m (\mathbf{h}_{i_m} \odot \mathbf{r} \odot \mathbf{h}_{j_m}), \quad (5)$$

where \odot is element-wise multiplication, $r \in \mathbb{R}^{d_{\text{gnn}}}$ is a parameter vector, $\mathbf{h}_i, \mathbf{h}_j \in \mathbb{R}^{d_{\text{gnn}}}$ are embeddings of the source and target nodes, respectively.

4.4 Model Training

We train PAIGE on the link prediction task using binary cross-entropy loss with negative sampling. We sample N negative targets for each observed triple in the training set. By sharing the negative samples within each batch of size B , we obtain $N \times B$ negative targets for each positive triple.

The adjacency list and the feature matrix for the nodes reside in CPU memory due to their large memory footprint. We uniformly sample a fixed number of neighbors to convolve over in *eq. 2* to control the memory consumption. The training procedure employs multiple CPU processes for neighborhood sampling, subgraph construction, feature extraction, and negative sampling, which then feed the constructed mini-batches to GPUs running model computations in parallel.

Data Split	Train	Dev	Test	Human Annotated
# of Examples	630K	89K	178K	1K

Table 1: Data summary.

Model	Precision@N (%)		
	P@1	P@5	P@10
RoBERTa	28.3	47.1	54.5
PAIGE-BoW	34.9	57.3	65.0
PAIGE-BiGRU	36.9	59.0	66.6
PAIGE (BiGRU \parallel GNN)	40.8	64.5	71.8

Table 2: PAIGE outperforms the RoBERTa-based baseline. The PAIGE-BoW and PAIGE-BiGRU remove the concatenation of the representations from the text and graph encoders; PAIGE-BoW uses Bag-of-Words encoder instead of BiGRU.

We compute RoBERTa embeddings for historical utterances offline and place them in Redis in-memory data store. The in-memory storage provides the CPU workers with fast access to the input features for a minibatch. This allows us to avoid repeated computation of utterance embeddings needed to produce the initial representations for the user and entity nodes.

4.5 Inference

Our graph design offers highly efficient inference as the representations of nodes added for incoming utterances do not affect other nodes in the graph. For an utterance node created at runtime, the only outgoing edge is the self-loop, and the only incoming edge is from the author’s user node. Thus, we can cache the representations for the users’ nodes from each GNN layer and compute the convolutions in *eq. 2* only for the new utterance node. We cache the representations of the NLU-hypothesis nodes multiplied with the decoder’s relation parameter vector, r in *eq. 5*, and use efficient Maximum Inner Product Search to select the rewrite.

5 Experiments

Here we evaluate PAIGE and empirically validate its performance. We begin with a quantitative assessment on general QR, followed by an evaluation on personalized use-cases.

5.1 Experimental Setup

Given a defective query, our task is to retrieve relevant rewrites from a large pool of candidates. The embeddings inferred by a model for a given user’s

Model	User Index		Global Index		
	P@1	P@5	P@1	P@5	P@10
RoBERTa	78.0	98.2	28.9	47.0	54.27
PAIGE	82.1	98.4	43.41	66.5	73.6

Table 3: Rewrite precision (%) in a *mock* setting where all target rewrites are among the previous queries of the user who issued the query. Performance gains from our model increase when rewrite candidates are not limited to the user’s previous queries (*User Index*) and a *Global Index* storing queries from all users is used.

	RoBERTa	PAIGE
Precision@1	77.9%	79.4%

Table 4: Rewrite precision on the human annotated dataset with rewrite candidates confined to individual users’ past queries.

queries are evaluated on future rewrite actions of that user. Table 1 summarizes our datasets. We construct the datasets following the procedure detailed in Section 2.1 and Appendix A. The implementation details and hyperparameters are available in Appendix B. We follow previous works and evaluate models using Precision@N (P@N) metrics. The P@N measures if at least one rewrite among the first N retrieved candidates matches the target’s utterance or NLU hypothesis. We implement the retriever from Fan et al. (2021) as our baseline but replace the Deep Structured Semantic Models (DSSM; Huang et al., 2013) with a pre-trained RoBERTa-base encoder to make it stronger. The baseline neural encoder takes a query’s text as input and learns to minimize cosine distance between the embeddings of the query and the rewrite. The pre-trained model is fine-tuned on the dataset used to train PAIGE.

5.2 Results

Table 2 shows that PAIGE outperforms the baselines by 12.5–17.3%, indicating the efficacy of our personalized query embeddings. We observe the largest absolute gain of 17.3% for P@10, and the largest *relative* gain of 43.8% for P@1.

Query’s semantics and graph’s topology are complimentary. Ablation results in Table 2 show that feeding query embeddings from GNN to the decoder without first concatenating them with utterance representations from GRU results in a large drop in performance of up to $\sim 5.5\%$. Removing the concatenation step *and* replacing GRU with less

Model	Precision@N (Relative change %)		
	P@1	P@5	P@10
RoBERTa	25.7	47.3	55.7
PAIGE	28.5 (+10.9)	54.5 (+15.3)	63.0 (+13.1)

Table 5: Rewrite precision (%) on a dataset of query–rewrite pairs such that the rewrites do not appear in the user’s dialogue history.

expressive Bag-of-Words (BoW) encoder for textual input features decreases performance only by additional $\sim 2\%$. Finally, all graph-based models outperform the RoBERTa-based baseline.

PAIGE improves personalized QR. To evaluate how well representations from PAIGE reflect users’ proclivities, we evaluate models on a dataset of defective queries with rewrites that are among the user’s previous requests. We consider two settings: 1) limiting rewrite candidates to individual users’ past queries (*User Index*), and 2) using a global candidate index storing queries from all users (*Global Index*). The former setting is an easier task as the indexes for individual users typically contain ~ 100 utterances while the latter uses a global index that contains $\sim 4.5\text{M}$ unique requests mapping to $\sim 2.2\text{M}$ hypotheses.

Table 3 shows that both models work well when the index is confined to the user’s past queries. Notably, PAIGE offers nearly 4% higher P@1, opening promising avenues for future work on consolidating the retrieval and ranking steps into a single model. The performance gap increases dramatically when attempting rewrites from a global index storing queries from *all* users — *i.e.*, compared to RoBERTa, PAIGE improves P@1 by 14.4% and P@10 by 19.3% (Table 3, *Global Index*).

For a more comprehensive evaluation of PAIGE, we evaluate models on a test set consisting of query–rewrite pairs identified by human annotators (1K examples). As in the previous paragraph, the human annotated dataset consists of defective queries for which rewrites can be found among the individual users’ historical requests. We confined rewrite candidates to the user’s past queries (*User Index* setting). PAIGE achieves 79.4% P@1 on this test set compared to 77.9% from RoBERTa (Table 4).

PAIGE generalizes to unseen user preferences. To check if our model generalizes to unseen user preferences, we evaluate it on a set of query – rewrite pairs such that the rewrites do not appear in the user’s dialogue history. We use examples

from entertainment domains that contain entities like songs and movies and tend to reflect users' affinities. Table 5 shows that our model offers up to 15.3% relative precision improvement over the baseline in this setting. This result is noteworthy since most traffic comes from entertainment domains.

6 Related Work

Non-personalized QR. Several prior studies have investigated the QR problem in a non-personalized context. Statistical QR models have been deployed in Alexa (Ponnusamy et al., 2020) and Google voice search (Sodhi et al., 2021). In their seminal work, Ponnusamy et al. (2020) apply an Absorbing Markov Chain (AMC) model as a collaborative filtering mechanism to mine reformulation patterns from sequences of user queries. At inference, an exact text match with a defect query in the indexed offline triggers a rewrite to the corresponding reformulation. Although statistical QR models are efficient at inference, they are transductive — limited to a fixed set of utterances — and do not generalize to unseen queries. Building on the work of Ponnusamy et al. (2020), Yuan et al. (2021) replace the Markov Chain with a GNN to capitalize on the distributed query representations, however, their method is still transductive. To facilitate inference on unseen queries, Chen et al. (2020) train a RoBERTa (Liu et al., 2019) encoder on a QR corpus. Other than the lack of personalization, the main limitation of these NR methods is that they treat each interaction independently, with side information encoded implicitly in the model's parameters. Recent studies show that performance of such methods tends to suffer when inputs contain rare words (Schick and Schütze, 2020; Biś et al., 2021) or spurious patterns (McCoy et al., 2019) such as common misconceptions (Podkorytov et al., 2021). PAIGE, on the other hand, uses the rich connectivities within the interactions graph and KG to refine the query representations.

Personalized QR. Fan et al. (2021) propose a Neural Retrieval NR-based personalized QR system. Through A/B testing on Alexa traffic, they demonstrate that the personalized approach improves user satisfaction relative to the non-personalized baseline. Fan et al. (2021) build a unique index for each user from the user's personal query log. They also build a global index storing historical queries from all customers. For each index type, dedicated

neural encoders are trained to retrieve rewrite candidates, which are then ranked by an arbitration model. Cho et al. (2021) extend personalization to the ranker, providing it with user-specific features. As in the case of non-personalized NR, these models rely on user-agnostic query embeddings. In comparison, PAIGE selects rewrites using query representations that depend on users' prior experiences.

Graph Neural Networks. GNNs use input graph structures as computational architectures that aggregate neighborhood information to produce contextual representations for the nodes (Kipf and Welling, 2017; Schlichtkrull et al., 2018). In recommendation systems, GNNs operate on collaborative knowledge graphs that combine user-item interactions and structured knowledge (Wang et al., 2019b, 2020) to predict users' interests. These methods model the relations between interactions to learn from the customers' collective behaviors and alleviate issues caused by sparsity in interactions data (Wang et al., 2019b). While these studies tend to use bipartite graphs, PAIGE supports any graph structure. Other works use GNNs to model language and KGs together (Ghazvininejad et al., 2018; Talmor et al., 2019; Yang et al., 2020; Zhang et al., 2022). In comparison, PAIGE jointly models the language, knowledge, and user interactions.

7 Limitations

PAIGE enables rewrite retrieval from a global set of reformulation candidates but not all defects will be covered by the index. Considering this, a generative approach to the problem (Roshan-Ghias et al., 2020) offers an advantage but generative models pose quality control challenges in production systems, where issues like hallucinations (Lee et al., 2018) could have harmful effects.

8 Conclusion and Future Work

We put forward a graph-based framework for learning user affinities from their interactions with conversational AI agent. The proposed framework learns directly from user feedback and requires no human annotated data. Through extensive experiments on real-world conversations, we demonstrate that our proposed PAIGE improves the performance of QR systems and, as a result, reduces friction in users' interactions with the AI agent.

References

- Daniel Biś, Maksim Podkorytov, and Xiuwen Liu. 2021. Too much in common: Shifting of embeddings in transformer language models and its implications. In *NAACL*.
- Zheng Chen, Xing Fan, and Yuan Ling. 2020. Pre-training for query rewriting in a spoken language understanding system. In *ICASSP*.
- Eunah Cho, Ziyang Jiang, Jie Hao, Zheng Chen, Saurabh Gupta, Xing Fan, and Chenlei Guo. 2021. Personalized search-based query rewrite system for conversational ai. *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*.
- Xing Fan, Eunah Cho, Xiaojiang Huang, and Chenlei Guo. 2021. Search based self-learning query rewrite system in conversational ai. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2018. A knowledge-grounded neural conversation model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70.
- Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context-and content-aware embeddings for query rewriting in sponsored search. In *SIGIR*.
- Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei Guo. 2021. Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
- Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. [Learning deep structured semantic models for web search using clickthrough data](#). In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management, CIKM '13*, page 2333–2338, New York, NY, USA. Association for Computing Machinery.
- Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
- Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fanjjang, and David Sussillo. 2018. Hallucinations in neural machine translation. In *Interpretability and Robustness in Audio, Speech, and Language Workshop, (NeurIPS 2018)*.
- Yuan Ling, Benjamin Yao, Guneet Kohli, Tuan-Hung Pham, and Chenlei Guo. 2020. Iq-net: A dnn model for estimating interaction-level dialogue quality with conversational agents. In *Proceedings of KDD Workshop on Conversational Systems Towards Mainstream Adoption*.
- 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*.
- Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448.
- Maksim Podkorytov, Daniel Biś, and Xiuwen Liu. 2021. How can the [mask] know? the sources and limitations of knowledge in bert. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.
- Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya. 2020. Feedback-based self-learning in large-scale conversational ai agents. In *AAAI*.
- Alireza Roshan-Ghias, Clint Solomon Mathialagan, Pragaash Ponnusamy, Lambert Mathias, and Chenlei Guo. 2020. Personalized query rewriting in conversational ai agents. *ArXiv*, abs/2011.04748.
- Timo Schick and Hinrich Schütze. 2020. Bertram: Improved word embeddings have big impact on contextualized model performance. In *ACL*.
- M. Schlichtkrull, Thomas Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. *ArXiv*, abs/1703.06103.
- Sukhdeep Sodhi, Ellie Ka-In Chio, Ambarish Jash, Santiago Ontan'on, Ajit Apte, Ankit Kumar, Ayooluwakunmi Jeje, Dima Kuzmin, Harry Fung, Heng-Tze Cheng, Jonathan J. Effrat, Tarush Bali, Nitin Jindal, Pei Cao, Sarvjeet Singh, Senqiang Zhou, Tameen Khan, Amol Wankhede, Moustafa Alzantot, Allen Wu, and Tushar Chandra. 2021. Mondegreen: A post-processing solution to speech recognition error correction for voice search queries. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.
- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

- Yizhou Sun and Jiawei Han. 2013. Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2):20–28.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.
- Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019a. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.
- Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019b. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958.
- Ze Wang, Guangyan Lin, Huobin Tan, Qinghong Chen, and Xiyang Liu. 2020. *CKAN: Collaborative Knowledge-Aware Attentive Network for Recommender Systems*, page 219–228. Association for Computing Machinery, New York, NY, USA.
- Zhuoyi Wang, Saurabh Gupta, Jie Hao, Xing Fan, Dingcheng Li, Alexander Hanbo Li, and Chenlei Guo. 2021. Contextual rephrase detection for reducing friction in dialogue systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1899–1905, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Shiquan Yang, Rui Zhang, and Sarah Erfani. 2020. GraphDialog: Integrating graph knowledge into end-to-end task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1878–1888.
- Siyang Yuan, Saurabh Gupta, Xing Fan, Derek Liu, Yang Liu, and Chenlei Guo. 2021. Graph enhanced query rewriting for spoken language understanding system. In *ICASSP*.
- Wentao Zhang, Yuezihan Jiang, Yang Li, Zeang Sheng, Yu Shen, Xupeng Miao, Liang Wang, Zhi Yang, and Bin Cui. 2021. Rod: reception-aware online distillation for sparse graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2232–2242.
- Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. GreaseLM: Graph REASONing enhanced language models. In *International Conference on Learning Representations*.

A Data Collection

We first find two consecutive user utterances in which the first turn was defective and the second was successful. To this end, we use defect detection models from Ling et al. (2020) and Gupta et al. (2021), and rephrase detectors from Wang et al. (2021). Moreover, we gather rewrites based on n-best hypotheses from ASR. In particular, if two consecutive user utterances are submitted less than t seconds apart and the first turn’s ASR n -best ($n > 1$) is the same as the second turn’s ASR 1-best, we deem them rephrases. Finally, we apply rule-based filters to reduce noise in the data. We retain examples for which the edit distance between the two turns is less than d , and the time gap between the two turns is shorter than t seconds. We empirically set the $t = 45$, $d = 7$ to reduce noise and maintain the opportunity.

B Implementation Details

We implement PAIGE using Deep Graph Library (DGL) (Wang et al., 2019a). We set the GNN’s dimension ($d_{gnn} = 256$) and number of GNN layers ($L = 3$). We use the batch size ($B = 512$) and the number of negative samples per training example ($N = 32$). With the in-batch sharing of corrupted triples, this results in ($N \times B = 16384$) negatives per example. The parameters of the model are optimized by AdamW with weight decay $1e - 3$. We use a warmup procedure that linearly increases the learning rate from 0. to $1e - 3$ over the first 2000 training steps. Afterward, the learning rate decreases following the values of the cosine function. We use a dropout rate of 0.1 applied to each layer (Srivastava et al., 2014). The models are trained using eight GPUs (NVIDIA V100) with total memory of 256GB, which takes ~ 20 hours on average.

C Additional Results

In Table 6 we report examples of faulty queries for which PAIGE provides correct rewrites but the baseline is unable to correct the defects.

Defective Query	RoBERTa Rewrite	PAIGE Rewrite
play pop junior clean	play pop culture radio clean	play pop clean
play downtown baby	play bad romance	play down down baby
what is the weather in	what is the weather today	what is the weather in utah

Table 6: Examples of defective queries where PAIGE provides correct rewrites, but the baseline fails to correct the defects.

D Relationships in the Interactions Graph

We provide additional details on the node and relation types in PAIGE interactions graphs. Table 7 summarizes node counts and their respective features and feature encoders. Table 8 describes the main relations captured by the graph’s edges.

# of Nodes	Count [†]	Features	Encoder	End-to-End
Utterance	4.4M	Text	BiGRU	✓
Defect-Utterance	635K		+ Adapter	✓
NLU-Hypothesis	2.2M	Frequency & Defect-rate	MLP + Adapter	✓✓
User	400K	Mean embedding of user’s queries	RoBERTa	×
Entity	650K	Mean embedding of queries w/entity	+ Adapter	✓
Domain	38	Embedding Matrix	Embeddings	✓
Intent	6722		+ Adapter	✓

Table 7: Our node encoding strategy allows graph updates without model re-training, *e.g.*, for new knowledge or users. [†]The node counts do not include nodes added *on-the-fly* during evaluation.

Interaction	Linking Entities	Link Type	Description
Affinity	Entity-User	Directed	A user interacts with an entity
Authorship	User-Utterance	Directed	A user submitted an utterance
Realization	Utterance-Hypothesis	Directed	An utterance expresses the hypothesis
Rewrites	DefectUtt.-Hypothesis	Directed	A defect-uttr. maps to non-defective hypothesis
Slot-Value-Rel	Entity-Hypothesis	Bi-Directed	An entity appears in hypothesis
Domain of	Domain-Hypothesis	Bi-Directed	A domain appears in hypothesis
Intent of	Intent-Hypothesis	Bi-Directed	A intent appears in hypothesis
Attribute	entity-entity	Bi-Directed	A relationship between entities

Table 8: Relationships in PAIGE’s Interactions Graph.

Fast Vocabulary Transfer for Language Model Compression

Leonidas Gee
Expert.ai, Italy
lgee@expert.ai

Andrea Zugarini
Expert.ai, Italy
azugarini@expert.ai

Leonardo Rigutini
Expert.ai, Italy
and
University of Siena
lrigutini@expert.ai

Paolo Torrioni
Department of Computer Science
and Engineering,
University of Bologna
paolo.torrioni@unibo.it

Abstract

Real-world business applications require a trade-off between language model performance and size. We propose a new method for model compression that relies on vocabulary transfer. We evaluate the method on various vertical domains and downstream tasks. Our results indicate that vocabulary transfer can be effectively used in combination with other compression techniques, yielding a significant reduction in model size and inference time while marginally compromising on performance.

1 Introduction

In the last few years, many NLP applications have been relying more and more on large pre-trained Language Models (LM) (Devlin et al., 2018; Liu et al., 2019; He et al., 2020). Because larger LMs, on average, exhibit higher accuracy, a common trend has been to increase the model’s size. Some LMs like GPT-3 (Brown et al., 2020) and BLOOM¹ have reached hundreds of billion parameters. However, these models’ superior performance comes at the cost of a steep increase in computational footprint, both for development and for inference, ultimately hampering their adoption in real-world business use-cases. Besides models that only a few hi-tech giants can afford, like GPT-3, even smaller LMs with hundreds of million parameters could be too expensive or infeasible for certain products. For one thing, despite being tremendously cheaper than their bigger cousins, fine-tuning, deploying and maintaining large numbers of such models (one for each downstream task) soon becomes too expensive. Furthermore, latency and/or hardware requirements may limit their applicability to specific

use-cases. For all these reasons, significant efforts – in both academic and industry-driven research – are oriented towards the designing of solutions to drastically reduce the costs of LMs.

Recently, several attempts have been made to make these models smaller, faster and cheaper, while retaining most of their original performance (Gupta et al., 2015; Shen et al., 2020). Notably, Knowledge Distillation (KD) (Hinton et al., 2015) is a teacher-student framework, whereby the teacher consists of a pre-trained large model and the student of a smaller one. The teacher-student framework requires that both the teacher and the student estimate the same probability distribution. While the outcome is a smaller model, yet, this procedure constrains the student to operate with the same vocabulary as the teacher in the context of Language Modeling.

In this work, we explore a method for further reducing an LM’s size by compressing its vocabulary through the training of a tokenizer in the downstream task domain. The tokenizer (Sennrich et al., 2016; Schuster and Nakajima, 2012; Kudo and Richardson, 2018) is a crucial part of modern LMs. In particular, moving from word to subword-level, the tokenization solves two problems: vocabulary explosion and unknown words. Moreover, the capability to tokenize text effectively in any domain is key for the massive adoption of pre-trained general-purpose LMs fine-tuned on downstream tasks. Indeed, tokenizers are still able to process out-of-distribution texts at the cost of producing frequent word splits into multiple tokens.

However, the language varies significantly in vertical domains or, more generally, in different topics. Hence, ad-hoc tokenizers, trained on the domain statistics, may perform a more efficient to-

¹<https://bigscience.huggingface.co/blog/bloom>

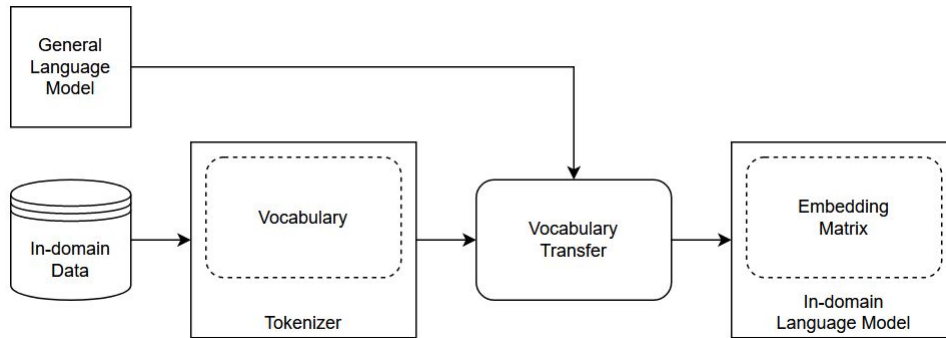


Figure 1: Sketch of the VT procedure. First, the vocabulary is constructed on the in-domain data, then an embedding is assigned to each token, transferring information from the pre-trained representations of the general-purpose language model.

kenization, reducing on average the length of the tokenized sequences. This is important since compact and meaningful inputs could reduce computational costs, while improving performance. Indeed, memory and time complexity of attention layers grows quadratically with respect to the sequence length (Vaswani et al., 2017). Furthermore, a vertical tokenizer may require a smaller vocabulary, which also affects the size of the embedding matrix, hence further reducing the model’s size.

Following this intuition, we propose a Vocabulary Transfer (VT) technique to adapt LMs to in-domain, smaller tokenizers, in order to further compress and accelerate them. This technique is complementary to the aforementioned model compression methods and independent of the type of tokenizer. As a matter of fact, we apply it in combination with KD.

Our experiments show that VT achieves an inference speed-up between $\times 1.07$ and $\times 1.40$, depending on the downstream task, with a limited performance drop, and that a combination of VT with KD yields an overall reduction up to $\times 2.76$.

The paper is organized as follows. After reviewing related works in Section 2, we present the methodology in Section 3, we then outline the experiments in Section 4 and draw our conclusions in Section 5.

2 Related Works

The goal of Model Compression is to shrink and optimize neural architectures, while retaining most of their initial performance. Research on LM compression has been carried out following a variety of approaches like quantization (Gupta et al., 2015; Shen et al., 2020), pruning (Zhu and Gupta, 2017; Michel et al., 2019) knowledge distillation (Sanh

et al., 2019; Jiao et al., 2020; Wang et al., 2020), and combinations thereof (Polino et al., 2018).

A most popular distillation approach in NLP was proposed by Sanh et al. (2019). The obtained model, called DistilBERT, is a smaller version of BERT, with the same architecture but half the layers, trained to imitate the full output distribution of the teacher (a pre-trained BERT model). DistilBERT has a 40% smaller size than BERT and retains 97% of its language understanding capabilities. This enables a 60% inference-time speedup. Further compression was achieved by Jiao et al. (2020) by adding transformer-layer, prediction-layer and embedding-layer distillation. The resulting model, TinyBERT, is 10 times smaller than BERT, with only four layers and reduced embeddings sizes. Related methods were proposed (Sun et al., 2020; Wang et al., 2020), achieving similar compression rates. All these works focus on the distillation of general-purpose language models. Gordon and Duh (2020) investigated the interaction between KD and Domain Adaptation.

Little focus has been devoted thus far to the role of tokenization in the context of model compression. Even in domain adaptation (Gordon and Duh, 2020), the vocabulary was kept the same. Both the versatility of the subword-level tokenization, and the constraints imposed by the teacher-student framework (same output distribution), discouraged such investigations. Recently, Samenko et al. (2021) presented an approach for transferring the vocabulary of an LM into a new vocabulary learned from new domain, with the purpose of boosting the performance of the fine-tuned model. To the best of our knowledge, we are the first to study VT in the scope of model compression.

3 Vocabulary Transfer

Let us consider a LM, trained on a general-purpose domain \mathcal{D}_{gen} and associated with a vocabulary \mathcal{V}_{gen} . Such a vocabulary is used by the LM’s tokenizer in order to produce an encoding of the input string via an embedding matrix E_{gen} defined on \mathcal{V}_{gen} . More specifically, a tokenizer is a function that maps a textual string into a sequence of symbols of a given vocabulary \mathcal{V} . Let \mathcal{T} be a tokenizer associated with a vocabulary \mathcal{V} and a string s , we have $\mathcal{T} : s \rightarrow (t_1, \dots, t_n), t_i \in \mathcal{V}, \forall i = 1, \dots, n$. Hence, the vocabulary of the tokenizer determines how words in a text are split, whether as words, sub-words, or even characters. These symbols, which define the LM’s vocabulary, are statistically determined by training the tokenizer to learn the distribution of a dataset.

Now, let us consider a vertical domain \mathcal{D}_{in} , also referred as *in-domain*. For the reasons discussed earlier, a vocabulary \mathcal{V}_{in} specialized on \mathcal{D}_{in} itself better fits the language distribution than \mathcal{V}_{gen} . Unfortunately, with a new vocabulary, embedding representations associated with the tokens of \mathcal{V}_{gen} would be lost. Thus, VT aims to initialize \mathcal{V}_{in} by re-using most of the information learned from the LM pre-trained on \mathcal{D}_{gen} . Once the new tokenizer \mathcal{T}_{in} has been trained on the in-domain dataset \mathcal{D}_{in} using a given vocabulary size, \mathcal{T}_{in} will be different from the LM’s tokenizer \mathcal{T}_{gen} . However, the two tokenizers’ vocabularies \mathcal{V}_{gen} and \mathcal{V}_{in} may still have a large portion of their symbols in common. Our objective is to transfer most of the information from \mathcal{V}_{gen} into \mathcal{V}_{in} . To this end, we first define a mapping between each symbol in \mathcal{V}_{in} and a set of symbols in \mathcal{V}_{gen} . Then, we define an assignment criterion, based on the mapping, to obtain the embeddings for the tokens of \mathcal{T}_{in} . The existence of a meaningful mapping is the only underlying assumption for VT, which holds for all the phonographic languages.

One such criterion, called Vocabulary Initialization with Partial Inheritance (VIPI), was defined by [Samenko et al. \(2021\)](#). Whenever a token is in \mathcal{V}_{in} but not in \mathcal{V}_{gen} , VIPI calculates all the partitions of the new token with tokens from \mathcal{V}_{gen} , then takes the minimal partitions and finally averages them to obtain an embedding for the new token. Differently, we define a simplified implementation of VIPI called FVT for Fast Vocabulary Transfer. Instead of calculating all tokenizations, FVT uses a straightforward assignment mechanism, whereby

Input: He was initially treated with interferon alfa.
\mathcal{T}_{gen} : He, was, initially, treated, with, inter,##fer,##on, al, ##fa, .
\mathcal{T}_{100} : He, was, initially, treated, with, interferon, alfa, .

Figure 2: Example of different tokenizations using a pre-trained or an adapted tokenizer. In the latter case, domain-specific words are not broken down into multiple word pieces.

Dataset	\mathcal{T}_{gen}	\mathcal{T}_{100}	\mathcal{T}_{75}	\mathcal{T}_{50}	\mathcal{T}_{25}
ADE	31	21	22	23	26
LEDGAR	155	131	131	132	135
CoNLL03	19	17	17	18	20

Table 1: Average sequence length on the three datasets with different tokenizers. \mathcal{T}_{gen} is the generic tokenizer (BERT cased), the same in each corpus, while $\mathcal{T}_{\%}$ are the tokenizers trained in the vertical domain itself, where % indicates the percentage of the original vocabulary size that has been set for training it.

each token $t_i \in \mathcal{V}_{in}$ is partitioned using \mathcal{T}_{gen} . If t_i belongs to both vocabularies, $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$, then $\mathcal{T}_{gen}(t_i) = t_i$ and the in-domain LM embedding $E_{in}(t_i)$ is the same as the embedding in the general LM:

$$E_{in}(t_i) = E_{gen}(t_i). \quad (1)$$

If instead $t_i \in \mathcal{V}_{in} \setminus \mathcal{V}_{gen}$, then the in-domain embedding is the average of the embeddings associated with the tokens produced by \mathcal{T}_{gen} :

$$E_{in}(t_i) = \frac{1}{|\mathcal{T}_{gen}(t_i)|} \cdot \sum_{t_j \in \mathcal{T}_{gen}(t_i)} E_{gen}(t_j). \quad (2)$$

Please notice that Equation 2 is a generalization of Equation 1. Indeed, in case $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$, Equation 2 falls back to Equation 1.

Once embeddings are initialized with FVT, we adjust the model’s weights by training it with MLM on the in-domain data before fine-tuning it on the downstream task. MLM eases adaptation and has already been found to be beneficial in ([Samenko et al., 2021](#)). We observed this trend as well during preliminary experiments, therefore we kept such a tuning stage in all our experiments.

As a baseline model, we also implement a method called Partial Vocabulary Transfer (PVT),

whereby only the tokens belonging to both vocabularies $t_i \in \mathcal{V}_{in} \cap \mathcal{V}_{gen}$ are initialized with pre-trained embeddings, while unseen new tokens are randomly initialized.

Transfer	ADE	LEDGAR	CoNLL03
\mathcal{T}_{gen}	90.80	80.93	89.43
$\mathcal{T}_{100} + \text{FVT}$	90.77	80.60	87.87
$\mathcal{T}_{75} + \text{FVT}$	90.40	80.93	87.90
$\mathcal{T}_{50} + \text{FVT}$	90.07	80.93	86.87
$\mathcal{T}_{25} + \text{FVT}$	90.27	81.03	86.17
$\mathcal{T}_{100} + \text{PVT}$	82.57	80.07	84.53
$\mathcal{T}_{75} + \text{PVT}$	82.47	80.33	84.63
$\mathcal{T}_{50} + \text{PVT}$	83.07	80.23	84.43
$\mathcal{T}_{25} + \text{PVT}$	83.57	80.20	83.47

Table 2: F1 results on the three benchmarks. A pre-trained language model fine-tuned on the task (\mathcal{T}_{gen}) is compared with models having differently sized in-domain tokenizers ($\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$) adapted by transferring information with FVT or PVT.

3.1 Distillation

VT can be combined with other model compression methods like quantization, pruning and KD. For some of the methods, the combination is trivial, since they have no impact on the vocabulary. KD, however, requires the vocabularies of the student and teacher to be aligned. Hence, its integration with VT is non-trivial. Accordingly, we set up a KD procedure with VT, in order to determine the effects of applying both VT and KD to an LM.

Our distillation consists of two steps. In the first step, we replicate the distillation process used in (Sanh et al., 2019) for DistilBERT, in which the number of layers of the encoder is halved and a triple loss-function is applied: a distillation loss, a MLM loss, and a cosine embedding loss. However, unlike the original setup, we do not remove the token-type embeddings and pooler. Inspired by Gordon and Duh (2020), after distilling the student on \mathcal{D}_{gen} , we further distil the student using \mathcal{D}_{in} . However, instead of adapting the teacher before the second distillation, we simply distil the student a second time on the in-domain dataset. Finally, we apply VT using either FVT or PVT and fine-tune the student model on the in-domain datasets.

Our choice of applying VT after KD is based on findings by Kim and Hassan (2020), that different input embedding spaces will produce different output embedding spaces. This difference in spaces is not conducive to knowledge transfer during dis-

Transfer	Distillation		
	ADE	LEDGAR	CoNLL03
\mathcal{T}_{gen}	90.47	78.37	86.90
$\mathcal{T}_{100} + \text{FVT}$	89.47	78.33	84.63
$\mathcal{T}_{75} + \text{FVT}$	88.57	78.90	84.23
$\mathcal{T}_{50} + \text{FVT}$	88.43	79.30	83.80
$\mathcal{T}_{25} + \text{FVT}$	88.23	78.10	83.13
$\mathcal{T}_{100} + \text{PVT}$	79.13	76.97	81.13
$\mathcal{T}_{75} + \text{PVT}$	78.87	76.93	81.40
$\mathcal{T}_{50} + \text{PVT}$	76.30	77.37	81.63
$\mathcal{T}_{25} + \text{PVT}$	77.90	77.33	79.50

Table 3: F1 results on the three benchmarks. A distilled language model fine-tuned on the task (\mathcal{T}_{gen}) is compared with models having differently sized in-domain tokenizers ($\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$) adapted by transferring information with FVT or PVT.

tillation. Hence, if VT were to be applied first to the student, its input embedding space would differ greatly from that of the pre-trained teacher during distillation.

4 Experiments

In the experiments we measure the impact of FVT on three main KPIs: quality (F1 score), size of the models and speedup in inference.

4.1 Experimental Setup

We consider for all our experiments the pre-trained cased version of BERT_{base} (Devlin et al., 2018) as our pre-trained language model. Its tokenizer is composed of 28996 wordpieces. We then define four vocabulary sizes for retraining our tokenizers. Specifically, we take the original vocabulary size and define it as a vocabulary size of 100%. We subsequently reduce this size to 75%, 50%, and 25%. From now on, we will refer to such tokenizers as $\mathcal{T}_{100}, \mathcal{T}_{75}, \mathcal{T}_{50}, \mathcal{T}_{25}$ respectively, while the original vocabulary will be called \mathcal{T}_{gen} .

Models are fine-tuned for 10 epochs with early stopping on the downstream task. We set the initial learning rate to $3 \cdot 10^{-5}$ and batch size to 64 for each task. The sequence length is set to 64 for ADE and CoNLL03 and 128 for LEDGAR. Each configuration is repeated 3 times with different random initializations. MLM is performed for one epoch.

4.2 Datasets

To best assess the effectiveness of VT, we apply it on three different tasks from three heterogeneous linguistic domains: medical (ADE), legal

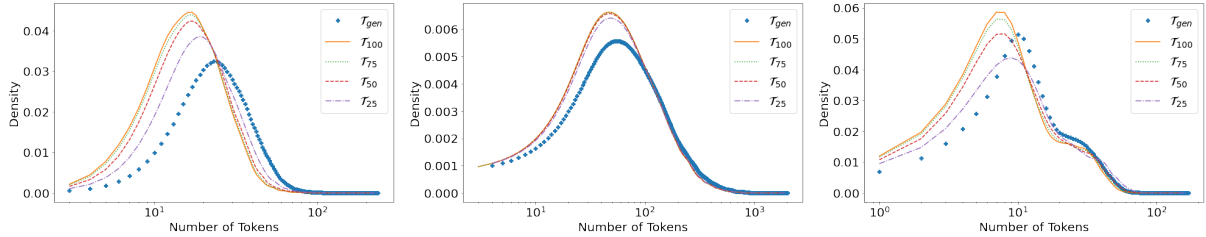


Figure 3: Sequence length distribution of each tokenizer on ADE, LEDGAR and CoNLL03 (left to right).

(LEDGAR) and news (CoNLL03). Table 4 reports the dataset statistics.

ADE. The Adverse Drug Events (ADE) corpus (Gurulingappa et al., 2012) is a binary sentence classification dataset in the medical domain. This domain is particularly suitable for investigating the benefits of VT, since documents are characterized by the presence of frequent technical terms, such as drug and disease names, that are usually rare in common language. Domain-specific words are usually split into multiple tokens, yielding longer sequences and breaking the semantics of a word into multiple pieces. An example is shown in Figure 2.

LEDGAR. LEDGAR (Tugener et al., 2020) is a document classification corpus of legal provisions in contracts from the US Securities and Exchange Commission (SEC). The dataset is annotated with 100 different mutually-exclusive labels. It is also part of LexGLUE (Chalkidis et al., 2022), a benchmark for legal language understanding.

CoNLL03. CoNLL03 (Tjong Kim Sang and De Meulder, 2003) is a popular Named Entity Recognition (NER) benchmark. It is made of news stories from the Reuters corpus. We chose this corpus because, differently from ADE and LEDGAR, the news domain typically uses a more standard language, hence we expect its distribution to differ less from the one captured by a general-purpose tokenizers in the web. Statistics in Table 1 confirms this hypothesis. We can observe that the sequence compression gain obtained with domain-specific tokenizers is less significant with respect to LEDGAR and ADE.

4.3 Results

We report an extensive evaluation of FVT on different setups and perspectives.

In-domain Tokenization. By retraining the tokenizer on the in-domain dataset, the average num-

Dataset	Train	Validation	Test
ADE	16716	3344	836
LEDGAR	60000	10000	10000
CoNLL03	14042	3251	3454

Table 4: Number of examples of each dataset.

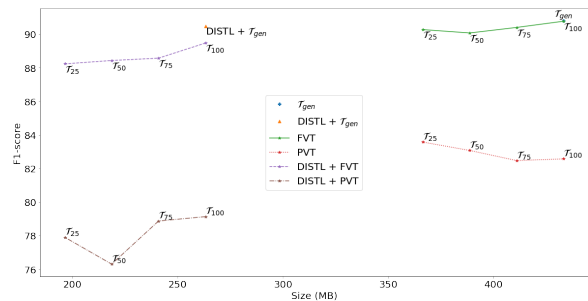


Figure 4: F1-score vs model size of VT with or without KD on ADE. VT and KD together can further compress a LM’s size in exchange for a limited performance drop. FVT is better than PVT. A smaller vocabulary size does not always imply a lower performance.

ber of tokens per sequence decreases since the learned distribution reduces the number of word splits, as shown in Table 1. In the medical domain, which is particularly specialized, we notice a remarkable 32% reduction of the average number of tokens per sequence. We expect this to yield a noticeable impact on inference time speedup. Furthermore, we can notice in Figure 3 that the sequence length distribution shifts to the left for the learned tokenizers. It can also be observed that by reducing the vocabulary size of the *in-domain* tokenizer, the sequence length distribution will begin to shift back to the right. Indeed, with fewer tokens in its vocabulary, the tokenizer will need to break down words more frequently into subwords.

Vocabulary Transfer. From the results shown in Tables 2 and 3, we note a few interesting findings. First, FVT vectors initialization method consistently outperforms the baseline PVT, which confirms the positive contribution of Equation 2. Sec-

Transfer	ADE			LEDGAR			CoNLL03		
	Δ F1	Δ Size	Speedup	Δ F1	Δ Size	Speedup	Δ F1	Δ Size	Speedup
\mathcal{T}_{gen}	90.80	433.32	1.00	80.93	433.62	1.00	89.43	430.98	1.00
\mathcal{T}_{100} + FVT	-0.04	0.00	1.40	-0.41	0.00	1.21	-1.75	0.00	1.07
\mathcal{T}_{75} + FVT	-0.44	-5.14	1.35	0.00	-5.14	1.21	-1.71	-5.17	1.07
\mathcal{T}_{50} + FVT	-0.81	-10.28	1.32	0.00	-10.27	1.10	-2.87	-10.33	1.02
\mathcal{T}_{25} + FVT	-0.59	-15.42	1.20	0.12	-15.41	1.09	-3.65	-15.50	0.99
Distil + \mathcal{T}_{gen}	-0.36	-39.26	1.97	-3.16	-39.24	1.97	-2.83	-39.48	1.95
Distil + \mathcal{T}_{100} + FVT	-1.47	-39.26	2.76	-3.21	-39.24	2.38	-5.37	-39.48	2.11
Distil + \mathcal{T}_{75} + FVT	-2.46	-44.40	2.64	-2.51	-44.37	2.38	-5.81	-44.64	2.11
Distil + \mathcal{T}_{50} + FVT	-2.61	-49.54	2.59	-2.02	-49.51	2.16	-6.30	-49.81	2.01
Distil + \mathcal{T}_{25} + FVT	-2.83	-54.68	2.37	-3.50	-54.64	2.14	-7.04	-54.98	1.96

Table 5: The first row (\mathcal{T}_{gen}) reports absolute values of the LM fine-tuned on the downstream task without VT or KD. The rows below show values relative to \mathcal{T}_{gen} .

ond, transferring vocabulary with FVT causes limited drops in performance, especially in LEDGAR (the largest one), where F1 slightly increases despite a 75% vocabulary reduction. We observed a higher degradation in CoNLL03. We believe this is due to the less specialized nature of the news domain, whereby the benefits of adapting the vocabulary to it are reduced. Overall, the effects of FVT on model performance do not have a steadily decreasing trend, as it might be presumed when reducing the vocabulary size, as also evident from Figure 4. In some cases, somewhat surprisingly, reducing the vocabulary size yields better model performance. In other cases, a 50% vocabulary size reduction yields better results than a full scale reduction or no reduction. Hence, vocabulary size should be considered as a hyper-parameter, where the model selection criteria may vary depending on the application’s KPIs, such as acceptable F1 drop, disk occupation and delay constraints.

Vocabulary Transfer and Distillation. The results summarized in Table 3 clearly indicate that KD is complementary to VT: there is no harm in applying them together, in terms of performance on the downstream task. Crucially, this guarantees a full exploitation of FVT in the scope of language model compression.

Compression and Efficiency. After showcasing that VT has limited impact on performance, we analyze and discuss its effects on efficiency and model compression. Table 5 reports the relative F1 drop on the downstream task with respect to the original LM (Δ F1), the relative reduction in model size (Δ Size) and the speedup gained by FVT alone

and by FVT combined with KD for varying vocabulary sizes. Either way, FVT achieves a remarkable 15%+ reduction with respect to BERT’s learnable parameters, with almost no loss in F1.

Furthermore, the reduced input length enabled by in-domain tokenization brings a reduction in inference time. The more a language is specialized, the higher is the speedup with in-domain tokenizers. This is also confirmed by the experiments, where the major benefits are obtained on the medical domain, with a x1.40 speedup. In CoNLL03 instead where language is much less specialized, speedup reduces and even disappears with \mathcal{T}_{25} . Distillation further pushes compression and speedup in any benchmark and setup, up to about 55% (of which 15% due to VT) and x2.75 respectively.

In summary, depending on the application needs, VT enables a strategic trade-off between compression rate, inference speed and accuracy.

5 Conclusion

The viability and success of industrial NLP applications often hinges on a delicate trade-off between computational requirements, responsiveness and output quality. Hence, language model compression methods are an active area of research whose practical ramifications are self-evident. One of the factors that greatly contribute to a model’s inference speed and memory footprint is vocabulary size. VT has been recently proposed for improving performance, but never so far in the scope of model compression. In this work, we run an extensive experimental study on the application of a lightweight method for VT, called FVT. An analysis conducted on various downstream tasks, application domains,

vocabulary sizes and on its possible combination with knowledge distillation indicates that FVT enables a strategic trade-off between compression rate, inference speed and accuracy, especially, but not only, in more specialized domains. Importantly, FVT appears to be orthogonal to other model compression methods.

In the future, we plan to fully integrate Vocabulary Transfer within Knowledge Distillation during the learning process in order to maximize the information transfer. We also plan to define a unified metric that combines all the KPIs, to facilitate model selection.

Acknowledgments

This work is part of the SCUDO project, which was funded by the "FESR 2014-2020" Regional Operational Program of the Tuscany Region (Italy), Call 2: "Research and development projects of the SMEs".

References

- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. [LexGLUE: A benchmark dataset for legal language understanding in English](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, Dublin, Ireland. 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](#).
- Mitchell Gordon and Kevin Duh. 2020. [Distill, adapt, distill: Training small, in-domain models for neural machine translation](#). In *Proceedings of the Fourth Workshop on Neural Generation and Translation*, pages 110–118, Online. Association for Computational Linguistics.
- Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR.
- Harsha Gurulingappa, Abdul Mateen Rajput, Angus Roberts, Juliane Fluck, Martin Hofmann-Apitius, and Luca Toldo. 2012. [Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports](#). *Journal of Biomedical Informatics*, 45(5):885 – 892. Text Mining and Natural Language Processing in Pharmacogenomics.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. DeBERTa: Decoding-enhanced BERT with disentangled attention. [arXiv preprint arXiv:2006.03654](#).
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. [arXiv preprint arXiv:1503.02531](#), 2(7).
- 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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.
- Young Jin Kim and Hany Hassan. 2020. [Fast-Formers: Highly efficient transformer models for natural language understanding](#). In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 149–158, Online. Association for Computational Linguistics.
- Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. [arXiv preprint arXiv:1808.06226](#).
- 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](#).
- Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.
- Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. [arXiv preprint arXiv:1802.05668](#).
- Igor Samenko, Alexey Tikhonov, Borislav Kozlovskii, and Ivan P Yamshchikov. 2021. Fine-tuning transformers: Vocabulary transfer. [arXiv preprint arXiv:2112.14569](#).
- 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](#).
- Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5149–5152. IEEE.

- 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.
- Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 8815–8821.
- Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. arXiv preprint arXiv:2004.02984.
- 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.
- Don Tuggener, Pius von Däniken, Thomas Peetz, and Mark Cieliebak. 2020. Ledger: A large-scale multi-label corpus for text classification of legal provisions in contracts. In Proceedings of the 12th Language Resources and Evaluation Conference, pages 1235–1241.
- 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.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Advances in Neural Information Processing Systems, 33:5776–5788.
- Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. arXiv preprint arXiv:1710.01878.

Multimodal Context Carryover

Prashan Wanigasekara, Nalin Gupta, Fan Yang, Emre Barut, Zeynab Raeesy, Kechen Qin, Stephen Rawls, Xinyue Liu, Chengwei Su, Spurthi Sandiri

Alexa AI-Natural Understanding, Amazon

{wprasha, nalgupta, fyaamz, ebarut, raeesyxr, qinkeche, sterawls, luxnyu, chengwes, spurthi}@amazon.com

Abstract

Multi-modality support has become an integral part of creating a seamless user experience with modern voice assistants with smart displays. Users refer to images, video thumbnails, or the accompanying text descriptions on the screen through voice communication with AI powered devices. This raises the need to either augment existing commercial voice only dialogue systems with state-of-the-art multimodal components, or to introduce entirely new architectures; where the latter can lead to costly system revamps. To support the emerging visual navigation and visual product selection use cases, we propose to augment commercially deployed voice-only dialogue systems with additional multi-modal components. In this work, we present a novel yet pragmatic approach to expand an existing dialogue-based context carryover system (Chen et al., 2019a) in a voice assistant with state-of-the-art multimodal components to facilitate quick delivery of visual modality support with minimum changes. We demonstrate a 35% accuracy improvement over the existing system on an in-house multi-modal visual navigation data set.

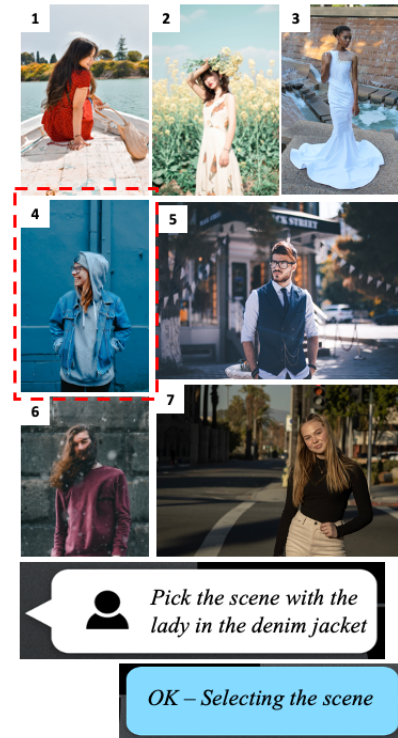


Figure 2: Scene Selection Use Case. The images are from unsplash.com and used here only for illustrative purposes.

1 Introduction

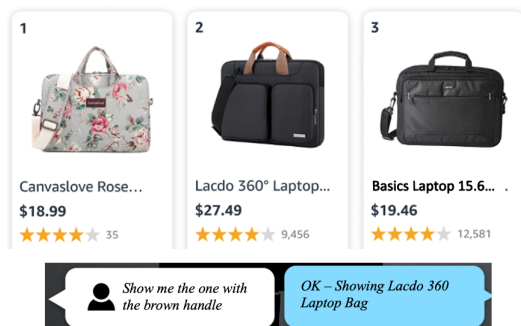


Figure 1: Product Selection Use Case

Tracking the state of the conversation and understanding context is a crucial component in voice-based dialogue systems. The Context Carryover



Figure 3: Video Selection Use Case

(CC) framework (Chen et al., 2019a; Naik et al., 2018; Sharaf et al., 2018; Rastogi et al., 2019) is a framework that handles identification and carryover of relevant context information in a multi-turn dialog interaction between a voice assistant and the user. The CC framework determines which tokens and intents in the most recent system-user interaction history are relevant as supporting information to fulfill user’s current request. The details

of the context carryover framework are well documented in (Chen et al., 2019a). One key limitation of the framework is that it is purely text-based, and therefore it struggles to capture user interactions that involve visual components. In this work, we introduce augmentations that enable the Context Carryover framework to deal with multimodal use cases. We focus on two specific use cases that’s related to a user’s visual navigation and selection experience: Visual product selection and visual scene and video selection.

Visual product selection, demonstrated in Fig. 1, consists of the use case where the user is referring to a single product on the screen. In the provided example, the user is shopping for a handbag and the voice assistant is displaying a number of handbags on the screen. The user selects one out of many handbags on the screen using a referring utterance, for instance the color of the handbag. The user is free to use any other natural language phrase that can differentiate the product from the others displayed on the screen.

In **visual scene and video selection**, as seen in Fig. 2, and Fig. 3, the user can refer to a scene or movie that has multiple products with a more cluttered visual landscape. Here, a “scene image” is defined as an image of an individual in a landscape wearing multiple products (dress, hat, purse, sunglasses etc.). The user then tries to select a scene using a referring expression (e.g., “The scene with the lady in the denim jacket”). In Fig. 3, a movie can be associated with multiple frames and the user can refer to the movie by referring to an action or content from a specific frame.

The main contributions of our work are:

1. We introduce a **Vision Augmentation** scheme that enables ingestion of visual content in a dialogue-based context carryover framework.
2. We introduce an **Aligned Vision and Text Augmentation** that incorporates the latest state-of-the-art developments in multi-modal contrastive learning to a dialogue-based context carryover framework.
3. The newly proposed methods result in significant accuracy improvements on an in-house data collected through Amazon Mechanical Turk (MTurk). We present sensitivity analyses that display the effectiveness of the various suggested model augmentations on our

in-house dataset.

4. We introduce a synthetic data generation pipeline that generates synthetic visual product selection data that helps to train the models and cuts down on manual annotation and MTurk survey costs.

2 Related work

Thanks to the advent of Transformer-based models over the past few years, multimodal representations have seen significant advances. The types of multimodal models can be roughly categorized into three, **a) Single Encoder** (Girshick et al., 2013; Long et al., 2014; Simonyan and Zisserman, 2014; Tan and Bansal, 2019; Chen et al., 2019b; Zhang et al., 2021; Li et al., 2020; Wanigasekara et al., 2022), **b) Dual Encoder** (Radford et al., 2021; Li et al., 2021a; Zhang et al., 2020; Jia et al., 2021; Yuan et al., 2021), and **c) Encoder-Decoder** models (Vinyals et al., 2014; Wang et al., 2021, 2022; Piergiovanni et al., 2022; Li et al., 2022). Attempts at unifying these foundational models have also been made in (Yu et al., 2022; Singh et al., 2021). **Single Encoder** models appear early in the multimodal literature and pave the way for the other 2 types of models. For this family of models, usually the image and text representations exist in separate spaces and there is an ensuing fusion layer. **Dual Encoder** models leverage image-text contrastive loss (Oord et al., 2018; He et al., 2019; Chen et al., 2020; Tian et al., 2019) during training, exhibit higher image-to-text alignment and bring the image, text representations to a common more aligned representation space. They perform well on image-text retrieval tasks but underperform in vision-language understanding tasks requiring higher reasoning, e.g., Visual Question Answering (VQA), and Natural Language Inference (NLI).

Our current task of Multimodal Context Carryover uses the latest advances in the multimodal representation learning space and injects state-of-the-art components with minimal changes into a framework for dialog tracking and slot selection, and results in a system that can handle multimodal user-system dialog interaction. Our current multimodal use cases are set up to be similar to a text-to-image retrieval task that occur within the context of a user-system dialog interaction. Thus, we incorporate the latest developments in **Dual Encoder** design in our work, since the approach is well suited for the multimodal text-to-image re-

trieval step. The latest state-of-the-art models that incorporate multimodal representations to dialogue state tracking systems, e.g., VDST (Pang and Wang, 2019), Flamingo (Alayrac et al., 2022), and VDTN (Le et al., 2022), would require costly system re-vamps.

Recently, Kottur et al. (2021) released a novel multimodal conversation dataset with labeled dialogue state (e.g., entity and dialogue act), which motivated further studies (Garcia et al., 2022; Agarwal et al., 2021). These datasets contain dialog act and products but do not have the scene and video information required for our purposes. For our current study we resort to collecting our own dataset through Amazon Mechanical Turk which is catered for our commercial needs.

3 Models

3.1 Problem Formulation

Each interaction between the user and the system can be formulated as a sequence of utterances H , consisting of alternating utterances between the user U and the system S : $H = (h_0^U, h_1^S, h_2^U, \dots, h_{dist}^{\{U,S\}})$, where each element $h \in H$ is an utterance either by the user, h^U or the system, h^S . We refer to H as the dialog history. A subscript *dist* denotes the utterance distance, which measures the offset from the most recent user utterance (h_0^U). The i^{th} token of an utterance with distance *dist* is denoted as $h_{dist}[i]$.

Each utterance in the dialog history, H consists of slots. A slot $x = (dist, k_x, v_x)$ in a dialog is defined as a key-value pair that contains information about an entity. For e.g., in Fig. 1 the user says, “Show me the one with the brown handle”. Here one of the slots would be [COLOR:Brown]. Each slot is defined by the distance of its corresponding utterance *dist*, slot key k_x and slot value v_x . We refer to X as the context slots which comprise of all the slots in the dialog history.

In addition to the context slots X which are derived from the dialogs, we also have on-screen lists which can be present in the current turn as shown in Fig. 1. Users can reference items in these lists either through visual features or through references to the title, e.g., “Canvaslove Rose one ...”. A list object $l = (k_l, v_l, I_l)$ in the current turn is defined as a key-value pair along with the associated image. The key k_l in our case is *ProductTitle*, the value v_l is the title itself and I_l refers to the image associated with the list object. We refer to L as all the list

objects in the current turn.

Given the dialog history H , context slots X and the on-screen list L , we can define the candidate slots as $C = (X \cup L)$. The task can be formulated as correctly identifying the subset of candidate slots C which are relevant to the current turn. A binary decision is made jointly over each of these candidate slots C by the model F_{joint} , which takes the slot interdependencies into consideration, i.e.,: $F_{joint}(C, H) = C_{carry}$, where $C_{carry} \subseteq C$.

The full details of the context carryover (CC) architecture which forms our baseline are provided in Appendix A.1. The baseline solution is not capable of ingesting visual content and hence cannot perform selection based on visual features. In Section 3.2, we introduce the vision and vision aligned text augmentations to the CC model, which add the capability to process visual features and are the main contributions of this paper.

3.2 Augmentations

3.2.1 Vision Augmentation

Recently CLIP (Radford et al., 2021), ALIGN (Jia et al., 2021), ALBEF (Li et al., 2021b), ConVIRT (Zhang et al., 2020) train dense, aligned image-text embeddings using contrastive loss. The training requires having N matched (*image, text*) pairs where the text can be free form. The bidirectional contrastive losses for the i^{th} image-text pair is given in equation 1 and equation 2 in Appendix A.2. The image and text are projected onto a shared embedding space $\mathbf{I} \in \mathbb{R}^d$, $\mathbf{T} \in \mathbb{R}^d$ respectively. $\langle \mathbf{I}_i, \mathbf{T}_i \rangle$ represents the cosine similarity and $\tau \in \mathbb{R}^+$ is a temperature parameter. The losses are then added as seen in equation 3 in Appendix A.2.

Our on-screen image selection use case is slightly different from this generic paired (*image, text*) training setting. In our case, the user makes a reference to a specific product, scene, or movie that is shown on the screen, which is more akin to a text-to-image retrieval task. The user also focuses on differentiating the desired product from the list of products shown on the screen. This is slightly nuanced than a generic text description of a product as the referring utterance is conditioned on the desired image and other surrounding images.

In our initial solution, we obtain the CLIP vision embeddings for the product images and add it to the CC framework as shown in Fig. 4a) which we term as the **Vision Augmentation**. In Fig. 4a), b), the term **List SeMI** stands for a List of [Se]mantically

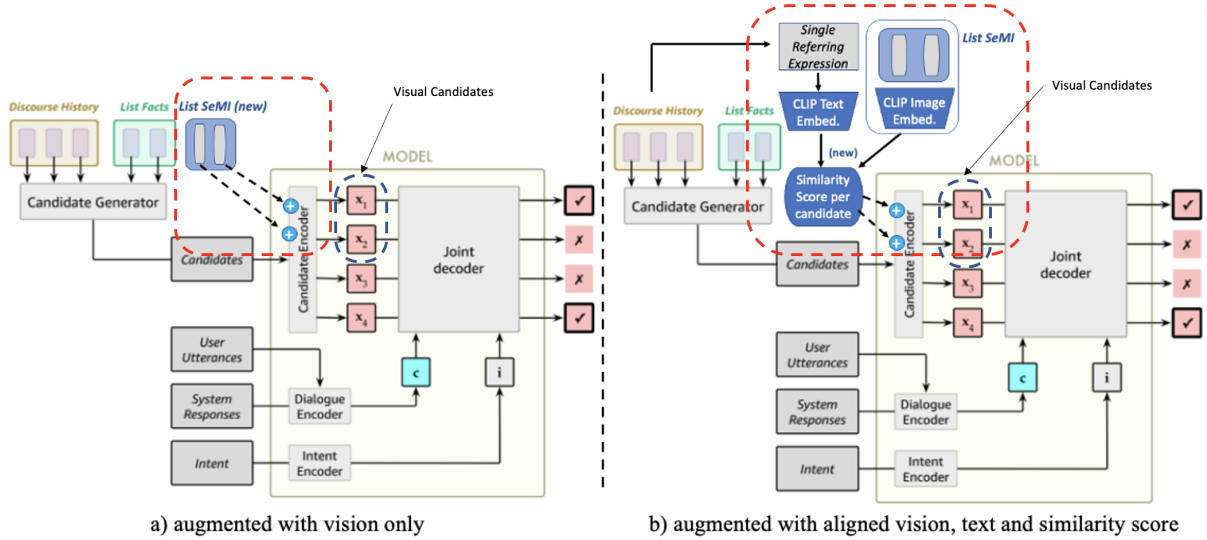


Figure 4: Augmented Context Carryover Models. Vision only augmentation (a), Vision, text and similarity score augmentation (b)

[M]eaningful [I]mages. More concretely, each product shown to the customer is represented by a list object $l = (k_l, v_l, I_l)$ and considered as a potential carryover candidate as mentioned in Section 3.1. As part of the Vision Augmentation, we use the CLIP visual embedding of the product image as I_l . These product list objects with CLIP visual embeddings are then sent to the CC Candidate Encoder. The CC Encoder-Decoder framework subsequently decides whether to carry over the list product object to the next dialogue state, which would signify a product selection. The rationale here is to simply augment the existing system with the visual modality and evaluate the effectiveness on a multimodal dataset. Even though we select CLIP as our initial vision embedding, the system is compatible with any embedding trained with a contrastive loss (e.g. we also show similar performance improvements on ALBEF (Li et al., 2021b)).

3.2.2 Aligned Vision and Text Augmentation

Using the notation from Section 3.1, at a given moment, there are n product list item objects $l = (k_l, v_l, I_l)$ shown to the user. Here, k_l is the key word *ProductTitle*, the value v_l is the actual title itself and I_l is the associated image. Given the product images $\{I_1, I_2, \dots, I_n\}$, their titles $\{v_1, v_2, \dots, v_n\}$, and the most recent user referring utterance h_0^U which is obtained from the dialog history, our task here is to find the best product list object l that matches the user’s request h_0^U . We utilize CLIP to bring images $\{I_1, I_2, \dots, I_n\}$ and the textual refer-

ring expression h_0^U to the same embedding space, and get the dot product similarity between each image in $\{I_1, I_2, \dots, I_n\}$ and h_0^U , as shown in Fig. 4b). In other words, we obtain the similarity score per candidate list image with the referring utterance. We term this as the **Aligned Vision and Text Augmentation**. The pseudocode for this operation is shown in Fig. 7, in Appendix A.3. The resulting multimodal dot product tensors are shown in Fig. 8, in Appendix A.4.

4 Experiments

4.1 Datasets

In this section, we describe the newly gathered Amazon Mechanical Turk (MTurk) multimodal dataset and the pre-existing text-only dataset.

4.1.1 Multimodal dataset

The MTurk dataset is created by showing Mechanical Turkers (MTurkers) product images, scene images, video thumbnails and asking them to pick one out of the many products, scenes or movies using a single referring utterance. We define one such referring act that includes multiple images and a single referring utterance as a single “instance”. The newly collected MTurk dataset has a Train/Dev/Test split sizes of 33,526/4,087/4,152 instances respectively. Additional details about the dataset are included in the Appendix A.5.

We also utilize an internally annotated dataset, that we refer to as the existing Context Carryover dataset, details of which can be seen in Appendix

Table 1: Results on the multimodal test set for various CLIP augmentation schemes and the CC baseline. The + <modality type> indicates an augmentation.

#	Visual Emb.	Model	Slot Level					
			Weighted (P, R, F1)				Accuracy	Δ
			P	R	F1	Δ		
1	None	Baseline	0.59	0.63	0.60	-	0.6301	-
2	CLIP	+ Utterance Text	0.65	0.67	0.54	-10.05%	0.6671	5.87%
3		+ Similarity	0.81	0.80	0.78	30.41%	0.7955	26.25%
4		+ Visual	0.78	0.78	0.78	31.12%	0.7811	23.96%
5		+ Visual + Utterance Text	0.79	0.78	0.79	32.26%	0.7844	24.49%
6		{+ Visual + Utterance Text + Similarity } (with non-contrastive fine-tuning)	0.80	0.80	0.79	33.35%	0.8003	27.01%
7		{ + Visual + Utterance Text + Similarity }	0.85	0.85	0.85	42.15%	0.8484	34.65%

A.6.

4.1.2 Synthetic Visual Data Generation Pipeline

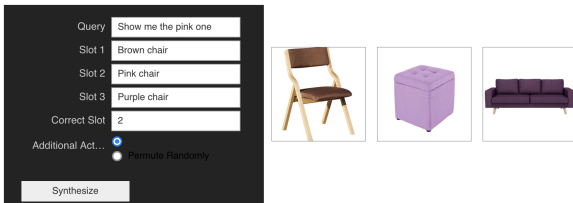


Figure 5: Synthetic Data Generation

Even though we use MTurk data for our current study, it is expensive to generate and infeasible to extend to more domains. An alternative cheaper and scalable approach to quickly collect carryover data in multimodal settings is to employ a data synthesizer. The synthetic data generation process is as follows; for each synthetic data sample, we first randomly sample a slot type (e.g., bag) and visual attributes (e.g., red) to create slot candidates. Note that for each slot, we only randomly select one type from a pre-defined *object* list, and we sample three different visual attributes under the same attribute category (e.g., color) from a pre-define *attribute* list. For example, we select one slot type bag from the object list, and then we draw three visual attributes red, blue, orange from the attribute list. We then combine them to obtain three slots: red bag, blue bag, orange bag to simulate the screen shown to the user when shopping for bags. In the second step, we retrieve an image from a product image catalog for each generated slot. To do so, we employ CLIP to get text embeddings of each slot: $\{e_1, e_2, e_3\}$. For each image in the product

catalog, we precompute the image embedding with the same CLIP model: $\{p_1, \dots, p_N\}$. We use the inner product as the similarity metric to perform the image retrieval: $I_x = \arg \max_i e_x^\top p_i$, where I_x denotes the image selected for slot x . To add more randomness, we retrieve the top- k similar images, and randomly select one image from the k candidates. After getting all of the product images (and associated metadata), we simulate a user selection phrase by randomly selecting one out of the three generated slots as the ground-truth and fill it in a predefined template.

4.2 Results

We compare various combinations of vision, text and similarity augmentation schemes against the baseline CC model in Table 1. All the models are trained on a combined CC and multimodal MTurk training dataset. Since we are interested in how our models perform on multimodal use cases, we show the results only on the multimodal test dataset. Further details of the experiment setup, such as model training details, are described in the Appendix A.7.

In Table 1, row 1 is the current baseline CC model, and rows 2-7 are the augmentations. Augmenting the existing CC framework with only CLIP Visual components (Table 1, row 4) gives a 23.96% accuracy improvement over the baseline on the multimodal test set. When adding CLIP Vision and CLIP user current utterance Text embeddings (Table 1, row 5), we see accuracy gains increase to 24.49%. The highest improvement comes when the CLIP vision, text embeddings and the dot product similarity scores (Table 1, row 7) are given to the CC framework with a 34.65% accuracy improvement over the baseline. These methods keep the

Table 2: Performance improvements on MTurk test data when MTurk training data is added to the CC train data set.

#	Model	Train Data	Slot Level					
			Weighted (P, R, F1)				Accuracy	Δ
			P	R	F1	Δ		
1	baseline	CC	0.53	0.67	0.53	-	0.67	-
2	baseline	MTurk	0.68	0.68	0.68	27.81%	0.68	2.34%
3	baseline	CC + MTurk	0.59	0.63	0.60	11.48%	0.63	-5.31%
4	Visual	CC + MTurk	0.78	0.78	0.78	46.17%	0.78	17.39%

CLIP embeddings frozen, but in Table 1, row 6 we attempt to fine-tune the CLIP embeddings in an end-to-end fashion using the CC framework. We find that fine-tuning the CLIP embeddings in our setting does not provide further gains. This maybe because the generic loss of the CC framework is non-contrastive (i.e., it’s cross-entropy based) and thus it does not improve the effectiveness of CLIP embeddings being further fine-tuned. A similar observation is recorded in parallel work Flamingo (Alayrac et al., 2022) and likened to “catastrophic forgetting”. In Table 1, row 3, we exclude the vision and textual embeddings altogether and provide only the dot product similarity scores. We find that only providing similarity scores (row 3) is on par with row 5 which is to provide both visual and textual embeddings. This has implications where the CC encoder can simply work with similarity scores instead of embeddings. Finally, in row 2 we only give the CC framework the CLIP Text embeddings (i.e., no vision components) and we find the performance to be worse than the other augmentations. We hypothesize that this is because the CC framework gets textual information from two sources in row 2. One from the dialogue history, which it processes in the usual fashion described in Section A.1 through the CC slot carryover framework, and one via the CLIP Text embeddings. Since there is no accompanying visual input, the CLIP Text input is redundant and might add additional noise, which leads to minor accuracy improvements and a weighted F1 degradation seen in row 2.

We are also interested in seeing how the CC baseline model behaves when the multimodal MTurk training data is added to its training set. More simply put, we want to check whether adding the MTurk training data to the CC framework will improve the performance of the CC baseline on the multimodal MTurk test set, *even if* the CC baseline is a purely text-based system that have no notion of the visual modality. From Table 2, row 1 we

see that even when no multimodal training data is added, the CC baseline still has an absolute accuracy of 67% on the MTurk test set. This can be attributed to the CC framework using the product image titles which are textual to make inferences. In Table 2, row 2 when the baseline is only trained on the multimodal MTurk dataset (without the 1.28M pre-existing CC dataset) there is an 2.34% accuracy improvement relative to the baseline, mainly due to training and testing distributions being similar. In Table 2, row 3, when the data sets are combined during training, the baseline shows a -5.31% degradation compared to Table 2, row 1 which indicates that the CC baseline is not equipped to handle a combined multimodal and non-multimodal dataset. In Table 2, row 4 we see that the best results are obtained when visual modality related model changes are added and the model is trained on the combined multimodal and pre-existing CC dataset.

We also experiment with ALBEF (Li et al., 2021b) embeddings and show results in Appendix A.8. We anticipate the science community to produce ever-improving dense multimodal embeddings as time goes on, and hope that our simple yet effective augmentation enables commercial frameworks to utilize the latest state-of-the-art embeddings with minimal changes.

5 Conclusion

We augment the existing Context Carryover framework with Visual and Vision Aligned Text components. We collect a multimodal dataset which mimics real world customer interactions to train and evaluate our models. We show a 35% accuracy improvement when the existing CC framework is augmented with Vision and Vision Aligned Text components.

Limitations

Our solution is only limited to the English language: our training data only contain products with English titles and all of the referring expressions are in English. Transferring the model to another language will require re-training the model and potential architecture changes. Furthermore, the Mturkers who provided the expressions in our study may not be representative of the user demographics, and the data may not provide a well grounded proxy of user behavior. While collecting more data can mitigate some of these limitations, curation and validation of visual expressions is a time-consuming and expensive process, which is why our dataset is limited in size.

Our models leverage visual embeddings from systems such as CLIP, ALBEF which have their own set of limitations. For instance, CLIP is known to fail in cases which requires counting objects, or relations of multiple objects in an image. Thus, visual models leveraging CLIP embeddings will have issues with referring expressions that refer to counts of objects. Further, we need to be cognizant and optimize for inference latency, which prevents us from using large scale language or vision models which could potentially improve upon the current solution.

Ethics Statement

Although our solution has no unethical applications or risky broader impacts, we need to consider aspects of fairness. In our setting, the images shown to the users can contain images of people along with the products. We need to consider how sensitive queries, e.g., ones that refer to protected attributes of the people in the image or expressions that contain hateful or derogatory speech, should be resolved.

During the data collection and model training process we take strong consideration on the type of referring expression we are curating and using to train the model. Expressions that contain references to protected and/or physical attributes of people are filtered out to ensure that our model is not capable of handling sensitive queries.

6 Acknowledgements

The authors would like to thank Balaji Kamakoti and Henry Zhang for their contributions in clearly defining the use cases and driving the visual product selection project. Rohit Parimi for managing

the engineering effort. Melanie Gens, Matt Johnson, and Adam Kalman for their contributions in engineering design. Chevanthie Dissanayake for her help in visualizations.

References

- Sanchit Agarwal, Jan Jezabek, Arijit Biswas, Emre Barut, Shuyang Gao, and Tagyoung Chung. 2021. [Building goal-oriented dialogue systems with situated visual context](#).
- Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. [Flamingo: a visual language model for few-shot learning](#).
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. [A simple framework for contrastive learning of visual representations](#).
- Tongfei Chen, Chetan Naik, Hua He, Pushpendre Rasgotgi, and Lambert Mathias. 2019a. [Improving long distance slot carryover in spoken dialogue systems](#).
- Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2019b. [Uniter: Universal image-text representation learning](#).
- 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](#).
- Jack FitzGerald, Shankar Ananthakrishnan, Konstantine Arkoudas, Davide Bernardi, Abhishek Bhagia, Claudio Delli Bovi, Jin Cao, Rakesh Chada, Amit Chauhan, Luoxin Chen, Anurag Dwarakanath, Satyam Dwivedi, Turan Gojavey, Karthik Gopalakrishnan, Thomas Gueudre, Dilek Hakkani-Tur, Wael Hamza, Jonathan J. Hüser, Kevin Martin Jose, Haidar Khan, Beiye Liu, Jianhua Lu, Alessandro Manzotti, Pradeep Natarajan, Karolina Owczarzak, Gokmen Oz, Enrico Palumbo, Charith Peris, Chandana Satya Prakash, Stephen Rawls, Andy Rosenbaum, Anjali Shenoy, Saleh Soltan, Mukund Harakere Sridhar, Lizhen Tan, Fabian Triefenbach, Pan Wei, Haiyang Yu, Shuai Zheng, Gokhan Tur, and Prem Natarajan. 2022. [Alexa teacher model](#). In [Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining](#). ACM.

- Francisco Javier Chiyah Garcia, Alessandro Suglia, José Lopes, Arash Eshghi, and Helen F. Hastie. 2022. [Exploring multi-modal representations for ambiguity detection & coreference resolution in the SIMMC 2.0 challenge](#). *CoRR*, abs/2202.12645.
- Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2013. [Rich feature hierarchies for accurate object detection and semantic segmentation](#).
- Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2019. [Momentum contrast for unsupervised visual representation learning](#).
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.
- Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. 2021. [Scaling up visual and vision-language representation learning with noisy text supervision](#).
- Satwik Kottur, Seungwhan Moon, Alborz Geramifard, and Babak Damavandi. 2021. [SIMMC 2.0: A task-oriented dialog dataset for immersive multimodal conversations](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 4903–4912. Association for Computational Linguistics.
- Hung Le, Nancy F. Chen, and Steven C. H. Hoi. 2022. [Multimodal dialogue state tracking](#).
- Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. [Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation](#).
- Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. 2021a. [Align before fuse: Vision and language representation learning with momentum distillation](#).
- Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq Joty, Caiming Xiong, and Steven Hoi. 2021b. [Align before fuse: Vision and language representation learning with momentum distillation](#).
- Xiujun Li, Xi Yin, Chunyuan Li, Pengchuan Zhang, Xiaowei Hu, Lei Zhang, Lijuan Wang, Houdong Hu, Li Dong, Furu Wei, Yejin Choi, and Jianfeng Gao. 2020. [Oscar: Object-semantics aligned pre-training for vision-language tasks](#).
- Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2014. [Fully convolutional networks for semantic segmentation](#).
- Chetan Naik, Arpit Gupta, Hancheng Ge, Mathias Lambert, and Ruhi Sarikaya. 2018. [Contextual slot carryover for disparate schemas](#). In *Interspeech 2018*. ISCA.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. [Representation learning with contrastive predictive coding](#).
- Wei Pang and Xiaojie Wang. 2019. [Visual dialogue state tracking for question generation](#).
- AJ Piergiovanni, Wei Li, Weicheng Kuo, Mohammad Saffar, Fred Bertsch, and Anelia Angelova. 2022. [Answer-me: Multi-task open-vocabulary visual question answering](#).
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. [Learning transferable visual models from natural language supervision](#).
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. [Language models are unsupervised multitask learners](#).
- Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Mathias Lambert. 2019. [Scaling multi-domain dialogue state tracking via query reformulation](#). In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 97–105, Minneapolis, Minnesota. Association for Computational Linguistics.
- Amr Sharaf, Arpit Gupta, Hancheng Ge, Chetan Naik, and Lambert Mathias. 2018. [Cross-lingual approaches to reference resolution in dialogue systems](#).
- Karen Simonyan and Andrew Zisserman. 2014. [Two-stream convolutional networks for action recognition in videos](#).
- Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. 2021. [Flava: A foundational language and vision alignment model](#).
- Hao Tan and Mohit Bansal. 2019. [Lxmert: Learning cross-modality encoder representations from transformers](#).
- Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2019. [Contrastive multiview coding](#).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).
- Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. [Show and tell: A neural image caption generator](#).

- Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. [Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework](#).
- Zirui Wang, Jiahui Yu, Adams Wei Yu, Zihang Dai, Yulia Tsvetkov, and Yuan Cao. 2021. [Simvlm: Simple visual language model pretraining with weak supervision](#).
- Prashan Wanigasekara, Kechen Qin, Emre Barut, Fan Yang, Weitong Ruan, and Chengwei Su. 2022. [Semantic vl-bert: Visual grounding via attribute learning](#). In [2022 International Joint Conference on Neural Networks \(IJCNN\)](#), pages 1–8.
- Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. 2022. [Coca: Contrastive captioners are image-text foundation models](#).
- Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luwei Zhou, and Pengchuan Zhang. 2021. [Florence: A new foundation model for computer vision](#).
- Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. [Vinvl: Revisiting visual representations in vision-language models](#).
- Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. 2020. [Contrastive learning of medical visual representations from paired images and text](#).

A Appendix

A.1 Existing Context Carryover framework

Our baseline architecture follows a similar approach to [Chen et al. \(2019a\)](#), where they jointly model the slots to make a slot carryover decision. One of the key differences in our model is that we act on entities extracted from an on-screen list along with the entities in the discourse history. We highlight the components of the existing CC model as follows:

Candidate Generation We create candidates based on a handcrafted slot map, which defines carryover compatibility between each pair of slots. We create a set of slots X from the context by leveraging the slot map to identify slots compatible with the current turn slots. We also append the candidates with the on-screen list entities.

Slot Encoder Given a candidate slot, which is represented as a (slotKey, slotValue), we average the word embeddings of the slot pair tokens and convert them into a fixed-length vector representation $\mathbf{x} \in R^{d_x}$.

Dialog Encoder We serialize the tokens in the dialog and use an LSTM ([Hochreiter and Schmidhuber, 1997](#)) to create a fixed length embedding. $\mathbf{c}_{dialog} = LSTM(H)$, where \mathbf{c}_{dialog} is the dialog encoding and H is the dialog.

Intent Encoder The intent tagged by an upstream Natural Language Understanding module is encoded by averaging the word embeddings of the tokens to create a fixed length embedding $\mathbf{int} \in R^{d_{int}}$.

Decoder Given the encoded representation of the slots $\{x_1, \dots, x_n\}$, dialog \mathbf{c}_{dialog} , and intent \mathbf{int} , we use the self-attention decoder presented in ([Chen et al., 2019a](#)). Self-attention allows the decoder to model relationships between all the slots in the dialog, which is shown to yield better results. In our model, we use 12 attention heads, which allows the model to jointly attend to information from different perspectives at different positions.

A.2 Contrastive Learning equations

$$\mathcal{L}_i^{(image \rightarrow text)} = -\log \frac{\exp(\langle \mathbf{I}_i, \mathbf{T}_i \rangle / \tau)}{\sum_{k=1}^N \exp(\langle \mathbf{I}_i, \mathbf{T}_k \rangle / \tau)}, \quad (1)$$

$$\mathcal{L}_i^{(text \rightarrow image)} = -\log \frac{\exp(\langle \mathbf{T}_i, \mathbf{I}_i \rangle / \tau)}{\sum_{k=1}^N \exp(\langle \mathbf{T}_i, \mathbf{I}_k \rangle / \tau)}. \quad (2)$$

The final loss is a weighted combination of the two losses averaged over the training dataset. Here $\lambda \in [0, 1]$ is a scalar weight.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left(\lambda \mathcal{L}_i^{(image \rightarrow text)} + (1 - \lambda) \mathcal{L}_i^{(text \rightarrow image)} \right). \quad (3)$$

A.3 Pseudocode for the vision aligned text dot product

The pseudocode for the vision aligned text dot product is shown in Fig. 7.

A.4 Multimodal dot product tensors

Fig. 8 a) shows the dot product between the utterance text and the product visual embeddings for visual product selection for each carryover candidate. Fig. 8 b) shows the dot product between the utterance text and scene or video and associated product’s visual embeddings for visual scene and video selection. Fig. 8 c) shows the dot product between the utterance text and the product metadata text associated with each candidate, where the product metadata can be available for both visual product selection and visual scene and video selection.

A.5 Additional Details on the multimodal dataset.

Some randomly sampled instances from the multimodal MTurk dataset for visual product selection are given in Fig. 6. To better simulate a real-world customer interaction with the voice assistant, the MTurkers are free to use any phrase to refer to the product image. Some MTurkers use specific product attributes like color, size, shape, product material, product label text while there are instances where more ambiguous terms are used (e.g., “*the animal one*”). To dissect the dataset further, we encode a few randomly selected product images and their associated labels in a joint CLIP embedding space.

As seen in Fig. 9a. The product images are shown on the horizontal axis and the product labels are shown on the vertical axis. The numbers in the table are CLIP similarity scores $\in [0, 1]$ between the images and the product labels (higher scores mean higher similarity). Fig. 9a has a dual purpose: first, it shows the products and their labels in a matrix format where products that match with

multiple labels or labels that match with multiple products can be clearly seen; second, it shows the effectiveness of CLIP embeddings in terms of quantifying image-text similarity. Ideally, the diagonal elements of the matrix should contain the largest scores, but we can see that there are a few off diagonal high similarity scores, which indicates that there is high ambiguity. In Fig. 9b we look at the alignment between the images, their labels and the referring utterance. It can be clearly seen from Fig. 9b row 1 that the image that matches the referring utterance (“*the black one*”) has the highest CLIP similarity score (the middle image has the highest alignment score; 0.24 with the referring utterance compared to the other two images in row 1).

A.6 Existing Context Carryover Dataset

The existing CC dataset is created by internal annotators who were shown the dialogue history, current turn, context slots and were asked to select all the appropriate slots for the current turn. The dialogs originate from a commercial voice assistant, and we process the data so that users are not identifiable (“de-identified”). The dataset spans 30 domains, 500 intents and includes both within domain (dialog that span a single domain) and cross domain cases (dialog than spans multiple domains). It has an average dialog distance length of 3.94 which is roughly 2 user turns and 2 system turns. The existing CC dataset has a Train/Dev/Test split size of 1,280,000/158,043/158,000 respectively.

A.7 Experimental Setup

We set the Context Carryover framework to the settings that are similar to the current commercial settings and run our experiments. The results in Table 1 and Table 2 use a context carryover threshold of 0.5. The context carryover threshold determines the probability threshold above which the slot will be labeled as a carryover instance (i.e., label of 1). We get the pretrained CLIP embeddings from the open-source CLIP (Radford et al., 2021) repo under the MIT license. For the CC model we use an embedding size of 300 for the dialog encoder and intent encoder. For the slot encoder, CLIP visual and CLP text we use an embedding size of 512. We use CLIP (ViT-B/32) (Dosovitskiy et al., 2020) as the vision encoder and a transformer (Vaswani et al., 2017) based text encoder as described in (Radford et al., 2019) for the CLIP text encoder. For the decoder, we use a single layer transformer based decoder with 12 attention heads which are



Figure 6: Randomly sampled instances from the MTurk dataset. Each instance is a single referring utterance and multiple associated product images. The label array signifies the ground truth label associated with the referring utterance. A label of 1 signifies the ground truth true label, and 0 otherwise.

```

encoded_list_images = clip_encode_list_images(list_images) # (batch x num_list_image_candidates x embedding_dim)
encoded_user_utterance_text = clip_encode_user_utterance(user_utterance_text), # (batch x 1 x embedding_dim)

normalized_list_images = normalize(encoded_list_images)
normalized_user_utterance_text = normalize(encoded_user_utterance_text)

similarity = dot.product(normalized_list_images, normalized_user_utterance_text) # (batch x num_list_image_candidates x 1)

```

Figure 7: Pseudocode for vision-text embedding dot product similarity

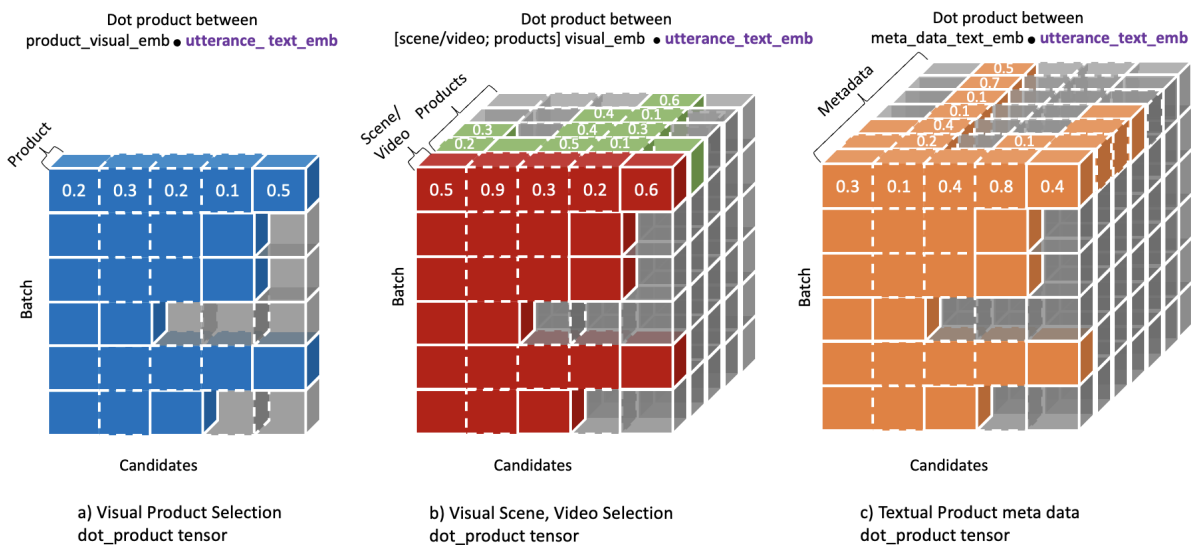


Figure 8: Tensors that contain the dot product between the user's referring utterance text and a) a candidate's single product visual embedding for the "Visual product selection" use case, b) a candidate's scene or video and multiple product visual embeddings for the "Visual scene and video selection" use case, c) a candidate's metadata text embeddings for both the use cases. Here, the metadata are the textual information that are associated with commercial products provided by sellers, marketplace annotators and at times generated by the system. The "candidates" here refer to the visual list item candidates.

then passed to a single layer feed-forward network to make binary decisions over the slots. The model is trained for 100 epochs using a batch size of 160 with an Adam optimizer and learning rate of 0.001. We train on a single p3.16xlarge instance, which consists of 8 GPUs.

A.8 ALBEF results

We also experiment with ALBEF (Li et al., 2021b) embeddings trained using a Large Language Model training framework (FitzGerald et al., 2022) and find them to have a similar performance to CLIP as seen in Table 3.

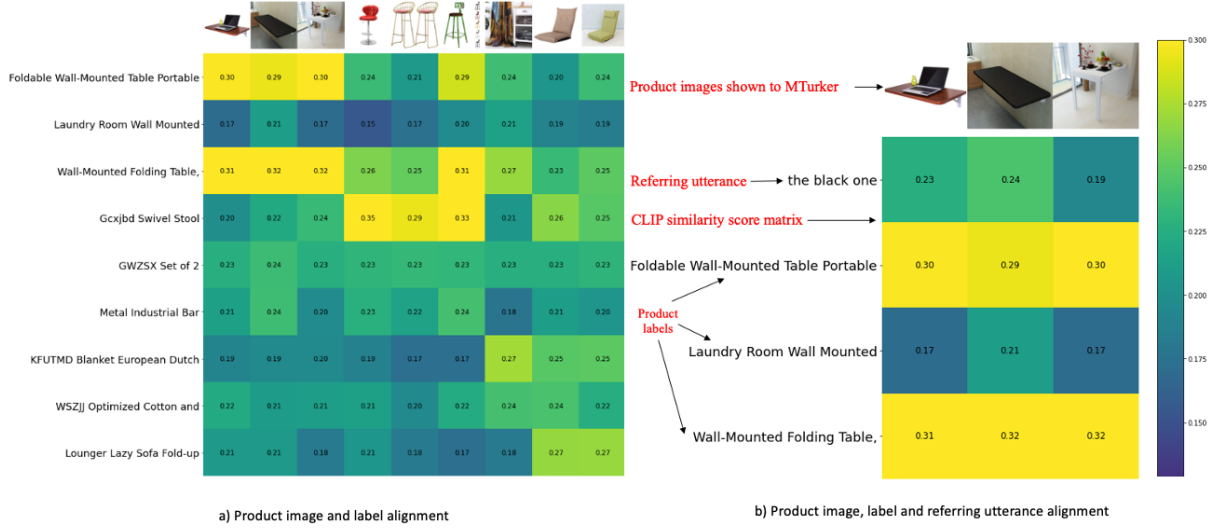


Figure 9: CLIP image-text alignment between images, labels, and referring utterance.

Table 3: Comparison with CLIP and ALBEF

#	Visual Emb.	Model	Slot Level					
			Weighted (P, R, F1)				Acc.	Δ
			P	R	F1	Δ		
1	None	Baseline	0.59	0.63	0.60	-	0.6301	-
2	CLIP	+ Visual	0.78	0.78	0.78	31.12%	0.7811	23.96%
3		+ {Visual + Utterance Text + Similarity}	0.85	0.85	0.85	42.15%	0.8484	34.65%
4	ALBEF	+ Visual	0.77	0.78	0.77	29.97%	0.7772	23.35%
5		+ {Visual + Utterance Text + Similarity}	0.86	0.86	0.86	44.34%	0.8617	36.76%

Distilling Multilingual Transformers into CNNs for Scalable Intent Classification

Besnik Fetahu, Akash Veeragouni, Oleg Rokhlenko, and Shervin Malmasi

Amazon.com Inc., Seattle, WA, USA

{besnikf, avveerag, olegro, malmasi}@amazon.com

Abstract

We describe an application of Knowledge Distillation used to distill and deploy multilingual Transformer models for voice assistants, enabling text classification for customers globally. Transformers have set new state-of-the-art results for tasks like intent classification, and multilingual models exploit cross-lingual transfer to allow serving requests across 100+ languages. However, their prohibitive inference time makes them impractical to deploy in real-world scenarios with low latency requirements, such as is the case of voice assistants.

We address the problem of cross-architecture distillation of multilingual Transformers to simpler models, while maintaining multilinguality without performance degradation. Training multilingual student models has received little attention, and is our main focus.

We show that a teacher-student framework, where the teacher’s unscaled activations (logits) on unlabelled data are used to supervise student model training, enables distillation of Transformers into efficient multilingual CNN models. Our student model achieves equivalent performance as the teacher, and outperforms a similar model trained on the labelled data used to train the teacher model. This approach has enabled us to accurately serve global customer requests at speed (18x improvement), scale, and low cost.

1 Introduction

For nearly all natural language understanding tasks, e.g. SuperGLUE (Wang et al., 2019), state-of-the-art results are obtained using pre-trained Transformer models. Their performance is dependent on their size and the amount of pre-training data, typically billions of tokens (Xue et al., 2021).

Intent Classification (IC), the task of understanding a user’s intent from an utterance, is a core component of all voice assistants such as Siri or Alexa. IC is challenging due to the hundreds of

intents and contexts that such systems must support, and IC performance has benefited greatly from Transformers (Chen et al., 2019). As voice systems have expanded support to new languages, the benefits of Transformers have multiplied with the advent of multilingual versions such as XLM-RoBERTa (Conneau et al., 2020).

Despite the advantages, deploying Transformers at scale is not always feasible, mainly due to: (i) large memory footprint (hundreds of GB),¹ and (ii) long inference time² that is prohibitive for applications processing millions of inputs per minute.

While approaches to reducing memory footprint — such as quantization (Vargaftik et al., 2021) or pruning (Gordon et al., 2020) — have been proposed, minimizing inference time is more challenging. Pruning can speed up inference, but there are limitations to how many self-attention layers can be pruned without loss of performance. *Knowledge Distillation* (KD) (Hinton et al., 2015) is another approach for transferring knowledge across model architectures, e.g. from Transformers to LSTMs (Wasserblat et al., 2020), to optimize performance.

However, cross-architecture distillation of multilingual Transformers to multilingual non-Transformer architectures has received almost no attention in the community. In this work we present the first exposition of this task. Specifically, we describe an approach used to deploy multilingual IC models for voice assistants allowing accurate inference at scale, speed, and low-cost.

We face two key challenges: (i) meeting *low inference* latency requirements, allowing us to globally serve customers in real time (millions per minute), and (ii) supporting *multi-linguality*, here we support 11 locales with 7 languages. Example utterances are shown below, which represent *e-commerce questions* issued in different languages.

¹e.g. GPT-3 (Brown et al., 2020) contains 175B parameters, roughly requiring 350GB, when using `float16`.

²Self-attention layers have quadratic time-complexity.

- how many calories are in a banana? (EN)
- wie viel fett enthält hühnchen? (DE)
- come si conservano le vongole in frigo (IT)
- cómo se hace un queque de yogur (ES)
- combien de temps peut-on réfrigérer une banane (FR)
- é possível congelar pastéis de nata (PT)

We use the teacher-student distillation paradigm, and show the optimal KD strategy for multilingual IC can leverage teacher logits alone (Mukherjee and Awadallah, 2020). Utterances for IC are typically 10-40 tokens, allowing us to exploit an efficient ConvNet architecture, and assess how they can obtain multilingual and pretrained knowledge from models like XLM-R via distillation.

While there have been previous attempts on distilling transformer models into ConvNets (Chia et al., 2019), our work is the first to explore cross-architecture multilingual KD on real-world applications with strict requirements for *latency* and *accuracy*. We make the following contributions.

- Knowledge distillation from Transformers to multilingual student (ConvNet) for intent classification based on the teacher-student paradigm;
- Minimal inference latency multilingual student models (18x speed up relative to teacher) without any loss in classification accuracy.
- Evaluation framework outlining the amount of distillation data required, and assessment of the student model’s generalization on unseen data.

2 Related Work

We now review some of the popular approaches for distilling and compressing Transformer models.

Model Finetuning. Eisenschlos et al. (2019) propose an efficient way to fine tune monolingual models on multilingual tasks by simply using the output of cross-lingual Transformer models as pseudo-labels. Their approach is based on the ULMFiT model (Howard and Ruder, 2018), where instead of the stacked LSTM networks (Hochreiter and Schmidhuber, 1997), they rely on quasi recurrent neural networks (Bradbury et al.) (QRNN). QRNN are similar to CNN, with the difference that the convolutional operators are done at each timestep, however, due to parallelization, they can be computed much more efficiently than LSTMs.

QRNNs are up to 16x faster than LSTMs, however, for our case, we find that ConvNets are more efficient than QRNNs, as they do not perform step-wise computations as QRNNs do. We compare the inference time of QRNNs and our proposed student model, and conclude that simple ConvNets have significantly lower inference time.

Model Compression. Ganesh et al. (2021) systematically review approaches for compressing transformers. To reduce memory usage, quantization is often applied (Vargaftik et al., 2021). Quantization reduces the amount of bits required to store network parameters. For example, parameters represented using `float32`, can instead be stored using only 16 or fewer bits, reducing memory usage significantly. This allows deploying larger models in compute infrastructure with limited resources.

Model pruning is a widely explored research direction for compression, mainly consisting of two techniques. First, in *unstructured pruning*, weights are zeroed out using different strategies (Gordon et al., 2020). Second, in *structured pruning* either the self-attention heads (Fan et al., 2019) or the encoder layers (Hou et al., 2020) are pruned.

Quantization and pruning facilitate usage of large transformers without the requirement of very high memory capacity (GPU or CPU) machines. Quantization, and unstructured model pruning, mainly reduce memory usage. Structured pruning, where encoder and self-attention layers are dropped, can improve efficiency. Yet, for many real-world applications the latency needs cannot be met (with few milliseconds, as is our case). For instance, pruning more than 50% of attention heads can lead to performance loss (Fan et al., 2019).

Knowledge Distillation (KD). Hinton et al. (2015) discuss the trade-offs between model size and performance. Training a larger model, and distilling its knowledge to a smaller model, either using the same training data or unsupervised training data, yields identical performance. The contrary cannot be said when training a small model directly, where the performance is significantly worse than its bigger counterpart. KD works under the *teacher-student* paradigm, where the teacher’s output is used to train the student model such that it mimics the teacher model in terms of the output.

There are several efforts in distilling transformers into recurrent (Wasserblat et al., 2020) and convolutional architectures (Chia et al., 2019). While recurrent models like LSTMs can significantly re-

duce memory footprint and latency, the step-wise sequential computation induces a large latency overhead that cannot be overcome. Conversely, ConvNets are highly efficient for text classification, both in terms of performance and latency.

Our approach is similar to that of Chia et al. (2019), in that we use CNNs as the main building block of the student model. However, we differ in several fundamental aspects and make contributions that further push the application of knowledge distillation. First, we deal with a multilingual task, which increases the complexity of the knowledge transfer from the teacher to the student model. Second, our ConvNet architecture is different to account for the multilingual requirement. Thirdly, we rely on unsupervised data for distillation, where we show how much data is necessary across different languages to have identical performance between the teacher and student models.

3 Multilingual Distillation Method

We now describe the KD approach: the IC task, the teacher/student models, and the learning objective.

3.1 IC Task

Our intent classification task requires categorizing utterances into two intents: *Commerce Question* (CQ), which are questions to the voice assistant about consumer products, and *Non-Commerce Question* (NCQ), which are all other questions.

3.2 Teacher and Student Models

Teacher Model: As our classifier is deployed globally in many languages, we use the multilingual XLM-RoBERTa (XLMR) transformer (Conneau et al., 2020) as our teacher model.

Given an utterance $\mathbf{w} = (w_1, \dots, w_n)$, consisting of n tokens, the teacher model is used to encode the input, $\mathbf{T}(\mathbf{w}) = \mathbf{h}_T(\mathbf{w})$, where $\mathbf{h}_T(\mathbf{w}) \in \mathbb{R}^m$ represents the [CLS] pooling representation from the last XLMR layer. This is fed to a softmax classification head, consisting of a dense projection that yields the raw activations of the network (i.e. unscaled log probabilities, or logits), which are then normalized to probabilities via softmax:

$$\text{logits}_T(\mathbf{w}) = \mathbf{h}_T(\mathbf{w})^T \cdot \mathbf{W}_T \quad (1)$$

$$p_T(\mathbf{w}) = \text{softmax}(\text{logits}_T(\mathbf{w})) \quad (2)$$

where $\mathbf{W} \in \mathbb{R}^{m \times C}$, C is the number of intent classes, and $\text{logits}_T(\mathbf{w})$ captures the intent of the utterance, and is used to student training.

Student Model: Figure 1 shows our student model architecture. We use a deep convolutional model (ConvNet) (LeCun and Bengio, 1995), which are widely used for text classification (Kim, 2014), mainly for two reasons. Firstly, their convolutional operators allow for effective extraction of local subword interactions in an utterance, allowing to connect question shapes (e.g. “how many calories”) and product names (e.g. *banana*). Secondly, convolutional operations can be computed in parallel, allowing for minimal inference time, an important prerequisite for real-world applications. Finally, as IC utterances are typically short (10-40 tokens), CNNs can sufficiently capture all the important local/global lexical cues for the IC task.

Tokenization and Word Representations: Utterances are tokenized using the byte-pair encoding tokenizer model (Sennrich et al., 2016). To create a multilingual ConvNet, we leverage pretrained multilingual subword embeddings (Heinzerling and Strube, 2018). This approach allows representations of all languages, with a small vocabulary.

Encoder: Five 1D kernels of size 2-6 tokens, each with 500 filters, are aggregated with max-pooling. The pooled outputs are concatenated to form the final text representation.

Next, the student model computes the utterance representation (cf. Figure 1 (e)), $\mathbf{S}(\mathbf{w}; \theta) = \mathbf{h}_S \in \mathbb{R}^m$, that is used to predict the intent probability:

$$\text{logits}_S(\mathbf{w}) = \mathbf{h}_S(\mathbf{w})^T \cdot \mathbf{W}_S \quad (3)$$

$$p_S(\mathbf{w}) = \text{softmax}(\text{logits}_S(\mathbf{w})) \quad (4)$$

where $\mathbf{W}_S \in \mathbb{R}^{m \times C}$ and θ represent the student model parameters that need to be optimized.

3.3 Distillation Learning Objective

We use soft targets from the teacher, i.e. the unscaled log probabilities prior to softmax normalization (the logits), to train the student. We directly supervise the training of the student model $\mathbf{S}(\mathbf{w}; \theta)$ such that $\text{logits}_S(\mathbf{w}) \approx \text{logits}_T(\mathbf{w})$.

To this end, our learning objective is to minimize the Mean Squared Error (MSE) loss over the logits (Mukherjee and Awadallah, 2020), computed over the N unlabelled instances:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \|\text{logits}_S(\mathbf{w}_i) - \text{logits}_T(\mathbf{w}_i)\|^2 \quad (5)$$

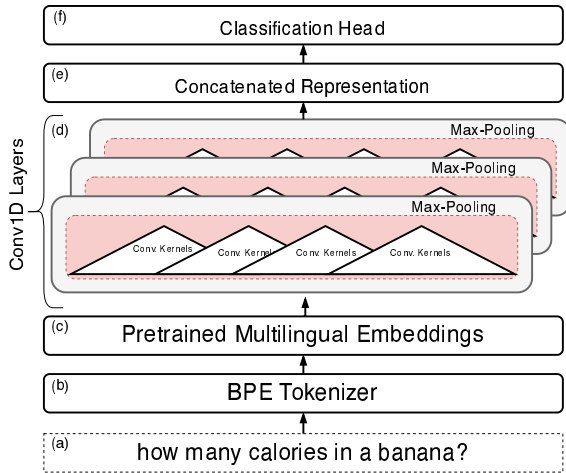


Figure 1: Student model: (a) input text, (b) byte-pair tokenizer, (c) pretrained embeddings compute subword representation; (d) 1D kernels with max-pooling; (e) concatenated representations; (f) output classification.

This logit loss encourages the student to output the same unnormalized activations as the teacher, which result in the same probabilities when normalized, and is more numerically stable to train. By minimizing \mathcal{L} on a large sample of unlabelled data, the distillation process can successfully transfer the intent classification knowledge from the teacher to the student. In this aspect, it is important to consider a *large* and *representative* sample given that \mathcal{L} can be minimal for a specific set of utterances, i.e. $|\text{logits}_S(\mathbf{w}) - \text{logits}_T(\mathbf{w})| < \epsilon$, however, for unseen utterances the difference between $|\text{logits}_S(\mathbf{w}) - \text{logits}_T(\mathbf{w})| \gg \epsilon$ (for some value that induces change in utterance’s label.)

4 Experimental Setup

We now describe the datasets used to train the teacher model, and for distillation. We also define the evaluation metrics used to assess how well the student model mimics its teacher.

4.1 Datasets

We use 3 types of data: (i) *teacher datasets* (supervised IC training data); (ii) *student datasets*, unannotated utterances to train the student; and (iii) *test data* used to evaluate the teacher and student.

Teacher Datasets: Table 1 (a) shows details of the training data used for the teacher model. Utterances come from 7 different languages and 11 locales. The task is imbalanced, but for confidentiality, the class distribution cannot be disclosed.

		(a)	(b)	(c)
Language	Locale	Teacher instances	Distillation instances	Test instances
English	en-US	1.7M	4M	733k
	en-GB	443k	32k	280k
Spanish	es-ES	7.5k	2.9M	106k
	es-US	6.8k	1.7M	11k
French	es-MX	6.8k	1.4M	61k
	fr-FR	7.3k	2.4M	62k
German	fr-CA	11.6k	911.3k	10.8k
	de-DE	1M	3M	208k
Italian	it-IT	7.4k	3M	11k
Portuguese	pt-BR	11k	1.3M	30k
Hindi	hi-IN	12.6k	647k	45k
		3.5M	22.6M	1.7M

Table 1: (a) Teacher data includes 3.5M utterances with annotated binary labels. (b) Student data has 22.6M unannotated utterances used to train the student model. (c) Test instances are used to evaluate both models.

Distillation Datasets: Table 1 (b) shows the statistics of the distillation data. We randomly sample a target number of utterances from each locale over a 1-month period. The data is unlabelled. Using unsupervised data allows the KD process to transfer any of Transformer’s pretrained knowledge that may not overlap with our supervised set.

Test Datasets Table 1 (c) shows the test datasets used to evaluate the performance of our teacher and student models. In total, our test set across all locales consists of 1.7M labelled instances.

4.2 Teacher and Student Configuration

Teacher Model: Model T is based on XLMR *base* model³ with a total of 278M parameters, is fine-tuned on data from Table 1 for our multilingual IC task. The model is trained by minimizing the cross-entropy loss function using the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of $lr = 3e - 5$.

Distilled Student Model: Model S is described in Figure 1, and consists of a total of 103M parameters. It is trained using the data in Table 1 (b) by minimizing the loss in Equation (5). A dropout rate of 10% is applied to the embeddings and CNN filters for regularization. We fine-tune the pretrained embeddings, and apply learning rate warmup over the first 2 epochs to prevent catastrophic forgetting. We train for 50 epochs (via the Adam optimizer), with an early stopping criterion of 3 consecutive epochs of non-decreasing loss.

³<https://huggingface.co/xlm-roberta-base>

4.3 Baseline

Our main objective is minimizing inference latency of Transformer models for IC. IC accuracy is not problematic for in-domain data, and most models achieve high performance (Larson et al., 2019).

QRNN: We focus in comparing only w.r.t the inference time between different approaches.⁴ We compare **S** to QRNN, proposed in (Bradbury et al.), and consider two configurations: (i) **QRNN₄**: with 4 ConvNet layers (as reported in Bradbury et al.), and (ii) **QRNN₅**: with 5 ConvNet layers, equivalent to the layers used in **S**.

Supervised Student Model: To assess whether distillation of teacher’s knowledge into **S** using unlabelled data is needed in the cases of abundance of labelled training data, we additionally train an identical model to **S** using the supervised training data in Table 1 instead, which we denote with **S_{sup}**. The training loss for **S_{sup}** is the cross-entropy loss.

4.4 Evaluation Metrics

Accuracy: We measure performance based on Precision (P) and Recall (R). Specifically, we compare the models at the threshold-agnostic P/R Break-Even Point (PR-BEP) (Joachims, 2005), where the precision and recall of the model are equal. To compare performance over all thresholds, we report PR-AUC (area under the PR Curve) which is a meaningful metric for imbalanced tasks (Liu et al., 2019). Due to confidentiality, we report only the gap of **S** and **S_{sup}** to **T**, as their absolute difference.

Efficiency: We measure wall-time t to compute the inference latency in *milliseconds* (ms). All measurements are the averaged latency over 100 trials, computed on an `m5.4xlarge` instance.⁵

5 Results

5.1 Model Accuracy

Overall Performance: Table 2 shows the performance difference between the teacher and student models. The overall PR-BEP gap across locales with 0.1% between **T** and **S** is negligible.⁶

Contrary to **S**, **S_{sup}** has a large gap to **T**, with an overall difference of 6%, and in certain locales, exceeding 30% in terms of PR-AUC. This gap

⁴Experimental evaluation shows that **S** achieves nearly identical performance to **T**. Thus, we do not report accuracy metrics for QRNN, given that its inference latency is higher.

⁵<https://aws.amazon.com/ec2/instance-types/>

⁶Statistically not significant per Binomial Proportion Test.

	S		S_{sup}	
	PR AUC	PR BEP	PR AUC	PR BEP
en-US	▼ 0.2%	▼ 0.2%	▼ 6%	▼ 6%
en-GB	▲ 0.2%	▲ 0.8%	▼ 5%	▼ 3%
es-ES	▲ 0.2%	▼ 0.1%	▼ 8%	▼ 6%
es-US	▼ 1.3%	▼ 0.9%	▼ 8%	▼ 7%
es-MX	▼ 0.4%	▼ 0.4%	▼ 10%	▼ 7%
fr-FR	▲ 0.1%	▲ 0.1%	▼ 8%	▼ 6%
fr-CA	▲ 0.4%	▲ 0.8%	▼ 7%	▼ 6%
de-DE	▼ 0.4%	▼ 0.6%	▼ 6%	▼ 5%
it-IT	▼ 0.3%	▼ 1.0%	▼ 11%	▼ 9%
pt-BR	▲ 0.2%	▼ 0.1%	▼ 33%	▼ 25%
hi-IN	▲ 0.2%	▼ 1.0%	▼ 30%	▼ 24%
average	▼ 0.1%	▼ 0.2%	▼ 6%	▼ 6%

Table 2: The gap between the student models (**S** and **S_{sup}**) to **T** is reported for the same test set. For **S** the gap of 0.2% is marginal, whereas for **S_{sup}** the gaps are highly significant according to the Binomial proportion test (p -value < 0.01).

remains large in the languages with the largest amount of supervised data, but is much more prominent in those with little data. This highlights two main findings: (i) **T** due to its Transformer architecture has a superior learning capacity when compared to directly training **S_{sup}** in a supervised manner; (ii) knowledge distillation allows us to successfully transfer the teacher’s pretrained knowledge to the student, allowing the student to acquire knowledge not present in the labeled data, and achieve similar generalization as the teacher (cf. Appendix A).

Incremental Distillation Performance: Table 3 shows the gap in performance between the student and teacher models, for varying amount of data used to train the student model. The data from Table 1 (b) is sampled using stratified sampling, with the locales representing the groups.

With only 1% of the data, the gap in terms of PR-BEP is 8.7% absolute points. Increasing this to 10% or more, the gap closes to less than 1%. Concretely, 10% represents 2.2M instances across all locales. In real-world settings it is reasonably cheap to obtain such amounts of unlabelled data.

Results indicate that with appropriate data, logit loss is highly effective for capturing the teacher’s knowledge. The student, using a different tokenizer and subword embeddings, is able to match teacher performance. Relative to other methods, logit loss is simpler to implement, and faster to train. For the IC task, we did not need to distill internal model

PR-BEP Absolute Percentage Difference to the Teacher Model.											
	1%	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
en-US	▼ 8.60%	▼ 2.70%	▼ 1.40%	▼ 0.90%	▼ 0.70%	▼ 0.30%	▼ 0.40%	▼ 0.20%	▼ 0.40%	▼ 0.60%	▼ 0.20%
en-GB	▼ 7.30%	▼ 2.10%	▼ 0.80%	▼ 0.50%	▼ 0.70%	▼ 0.10%	▼ 0.30%	0%	▲ 0.40%	▲ 0.20%	▲ 0.10%
es-ES	▼ 9.40%	▼ 4.40%	▼ 1.40%	▼ 0.70%	▼ 0.60%	▼ 0.80%	▼ 0.80%	▼ 0.90%	▼ 0.50%	▼ 0.70%	▼ 1.00%
es-US	▼ 9.00%	▼ 3.90%	▼ 3.60%	▼ 0.90%	▼ 0.60%	▼ 1.20%	▲ 0.60%	▲ 0.30%	▲ 0.90%	▲ 0.90%	▼ 1.20%
es-MX	▼ 9.80%	▼ 4.60%	▼ 1.90%	▼ 1.50%	▼ 0.80%	▼ 1.40%	▼ 0.80%	▼ 0.60%	▼ 0.40%	▼ 0.90%	▼ 1.20%
fr-FR	▼ 8.30%	▼ 3.10%	▼ 1.30%	▼ 1.60%	▼ 0.90%	▼ 1.10%	▼ 0.80%	▼ 0.70%	▼ 0.80%	▼ 1.00%	▼ 0.70%
fr-CA	▼ 9.50%	▼ 3.00%	▼ 0.30%	▼ 0.30%	▼ 0.30%	▲ 0.30%	▲ 1.20%	▼ 0.30%	▲ 0.60%	▼ 1.20%	▼ 0.30%
de-DE	▼ 8.20%	▼ 3.30%	▼ 1.70%	▼ 1.20%	▼ 1.30%	▼ 0.70%	▼ 0.60%	▼ 0.60%	▼ 0.60%	▼ 0.50%	▼ 0.20%
it-IT	▼ 11.20%	▼ 4.30%	▼ 2.00%	▼ 1.10%	▼ 1.90%	▼ 1.20%	▼ 0.70%	▼ 1.00%	▼ 1.10%	▼ 1.10%	▼ 1.40%
pt-BR	▼ 11.80%	▼ 5.10%	▼ 1.80%	▼ 1.70%	▼ 2.60%	▼ 1.00%	▼ 0.40%	▼ 0.80%	▼ 1.40%	▼ 0.60%	▼ 1.50%
hi-IN	▼ 11.90%	▼ 8.30%	▼ 5.90%	▼ 4.30%	▼ 3.10%	▼ 2.40%	▼ 1.80%	▼ 1.40%	▼ 1.20%	▼ 1.10%	▼ 1.30%
average	▼ 9.54%	▼ 4.1%	▼ 2.01%	▼ 1.34%	▼ 1.22%	▼ 0.95%	▼ 0.76%	▼ 0.62%	▼ 0.75%	▼ 0.8%	▼ 0.82%

Table 3: PR-BEP performance of the student model trained on varying portion of the distillation data from Table 1 (b). Overall, with 1% of the data used for distillation, the student model has an average gap in terms of PR-BEP of 8.7%. With increasing percentage of data used for distillation the gap is shrunk to 0.6% for 40% of the data.

values (e.g. representation loss). We did not use the supervised data for student training (e.g. with cross-entropy loss); our finding is that a sufficiently large and representative unsupervised sample will contain samples similar to those in the supervised set, as well as dissimilar ones, thus allowing the transfer of knowledge represented by both the labelled data and the Transformer’s pretrained knowledge.

5.2 Inference Latency

A drawback in deploying transformers is their prohibitive inference latency, mainly impacted by: (i) model size, and (ii) number of encoder layers.

Figure 2 shows the latency for different ablations of \mathbf{T} (with varying numbers of encoder layers), and the latency of \mathbf{S} , as the model with the lowest latency. Comparing \mathbf{T} and \mathbf{S} , our student model has nearly 18x lower inference latency, with only 2.7ms. This represents a drastic latency reduction, allowing us to process inputs extremely quickly.

For the teacher ablations, even for \mathbf{T}_1 , the inference latency is still higher than that of \mathbf{S} , with an additional +1.24ms latency per utterance. Furthermore, pruning layers is not lossless in terms of performance, especially in this case where only one layer is retained (Fan et al., 2019). The bottom part of Figure 2 shows the gap of the different pruned teacher models \mathbf{T}_l w.r.t the full model \mathbf{T} . The gap is high when we use fewer than 8 layers, with more than 12% drop in PR-BEP. It is clear that there is no clear trade-off between self-attention layer pruning and inference latency reduction.

Finally, comparing the baseline \mathbf{QRNN}_4 and \mathbf{QRNN}_5 , we note that the proposed student architecture, relying solely on ConvNets, results in a significantly lower inference latency. Our student

architecture has 3.8x and 2.95x lower latency than \mathbf{QRNN}_5 and \mathbf{QRNN}_4 , respectively. This significant increase in terms of latency can be explained by the fact that \mathbf{QRNN} applies its convolutional operators for timestep (each token in an utterance), which although more efficient than LSTMs (due to parallelization), it introduces a significant overhead over the traditional ConvNet architectures.

6 Conclusions

We described an approach for distilling knowledge from Transformer into a single multilingual CNN. To our knowledge this is the first detailed exposition of cross-architecture KD to multilingual student models. We leverage the outlined framework to accurately serve predictions for our customers at speed, scale, low-cost, and across all languages. Empirically we showed how such a KD framework can be utilized in practice:

1. With sufficient unsupervised data, leveraging logits is an optimal distillation strategy for training smaller and more efficient student models, without significant performance loss.
2. KD allows smaller and more efficient models to mimic the performance of their teacher counterparts, which is not the case if similar architectures are directly trained using labelled data.
3. KD is highly preferred over other techniques such as pruning. Transformers, even with a single encoder layer have higher inference latency, and the performance drop with pruning is large, where \mathbf{T}_1 has a 23% and 22% gap in terms of PR-BEP w.r.t \mathbf{T} and \mathbf{S} , respectively.

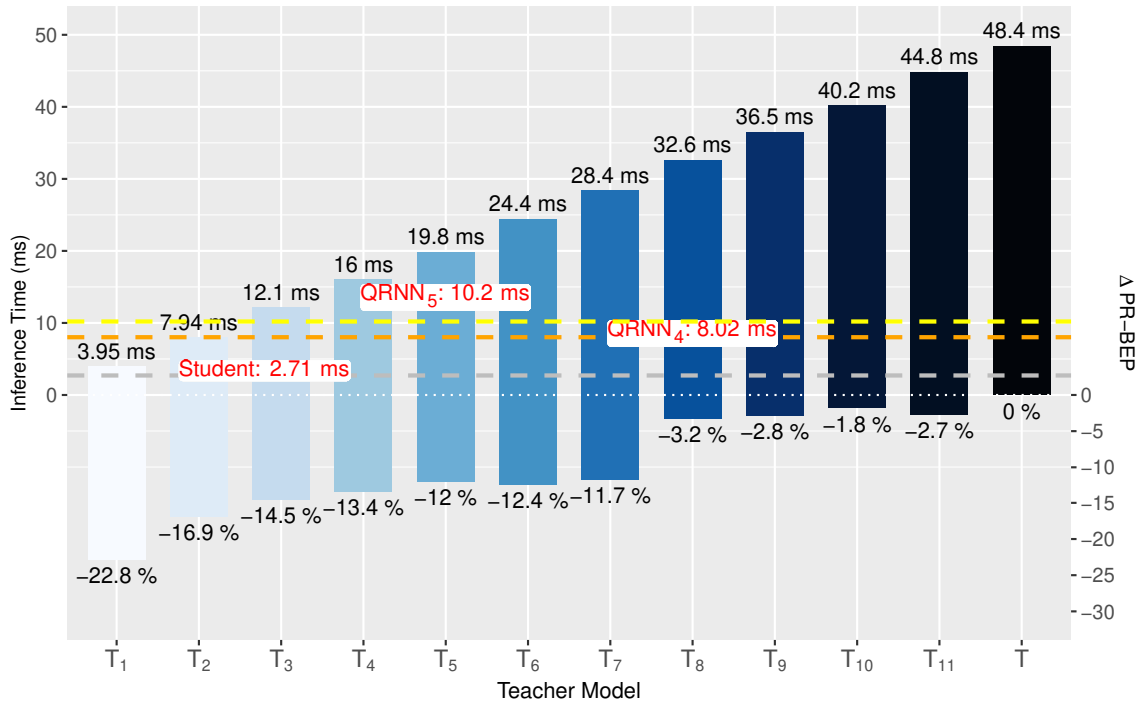


Figure 2: The upper plot shows the *inference latency* (in milliseconds) for the teacher models, where T_l ($l \in \{1, \dots, 11\}$). T_1 is a single encoder layer, with the other 11 layers pruned. The bottom plot shows the gap in terms of PR-BEP of the T_l models to the full teacher model. Note that, T_1 which has the closest inference time to S (with $2.71ms$ latency), has a 22.8% and 22.5% drop in terms of PR-BEP w.r.t T and S , respectively. Similarly, for QRNN, the inference time is shown in the orange and yellow dashed lines, with a latency of $QRNN_4 = 8.02 ms$ and $QRNN_5 = 10.02 ms$.

- For IC, a single multilingual CNN using multilingual subword embeddings can match the teacher performance despite using a different tokenizer. It is highly efficient, decreasing the latency by nearly 18x relative to the teacher.
- Using as few as 2-3M distillation instances, S achieves highly comparable performance as T , with less than 1% PR-BEP difference. The gap diminishes to 0.95% with just 40% of distillation data, specifically 8.8M instances.
- S achieves the same generalization power as T . On a held out test set (unseen during training and distillation), the output probabilities have a very low KL divergence (cf. Appendix A).

References

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

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](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Yew Ken Chia, Sam Witteveen, and Martin Andrews. 2019. [Transformer to CNN: label-scarce distillation for efficient text classification](#). *CoRR*, abs/1909.03508.

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, ACL 2020, Online, July 5-10, 2020*, pages 8440–8451. Association for Computational Linguistics.
- Susan Dean and Barbara Illowsky. 2018. Descriptive statistics: skewness and the mean, median, and mode. *Connexions website*.
- Julian Eisenschlos, Sebastian Ruder, Piotr Czapla, Marcin Kardas, Sylvain Gugger, and Jeremy Howard. 2019. **Multifit: Efficient multi-lingual language model fine-tuning**. 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 5701–5706. Association for Computational Linguistics.
- Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.
- Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. 2021. **Compressing Large-Scale Transformer-Based Models: A Case Study on BERT**. *Transactions of the Association for Computational Linguistics*, 9:1061–1080.
- Mitchell A Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing bert: Studying the effects of weight pruning on transfer learning. *arXiv preprint arXiv:2002.08307*.
- Benjamin Heinzerling and Michael Strube. 2018. **Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages**. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*. European Language Resources Association (ELRA).
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. **Dynabert: Dynamic BERT with adaptive width and depth**. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- 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.
- Thorsten Joachims. 2005. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384.
- Yoon Kim. 2014. **Convolutional neural networks for sentence classification**. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. **An evaluation dataset for intent classification and out-of-scope prediction**. 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 1311–1316. Association for Computational Linguistics.
- Yann LeCun and Yoshua Bengio. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.
- Shengchao Liu, Yingyu Liang, and Anthony Gitter. 2019. Loss-balanced task weighting to reduce negative transfer in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9977–9978.
- 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.
- Subhabrata Mukherjee and Ahmed Hassan Awadallah. 2020. **Xtremedistil: Multi-stage distillation for massive multilingual models**. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 2221–2234. Association for Computational Linguistics.
- 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.
- Shay Vargafik, Ran Ben Basat, Amit Portnoy, Gal Mendelson, Yaniv Ben-Itzhak, and Michael Mitzenmacher. 2021. **Communication-efficient federated learning via robust distributed mean estimation**. *CoRR*, abs/2108.08842.

- 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, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–3275.
- Moshe Wasserblat, Oren Pereg, and Peter Izsak. 2020. Exploring the boundaries of low-resource bert distillation. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 35–40.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. [mt5: A massively multilingual pre-trained text-to-text transformer](#). In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 483–498. Association for Computational Linguistics.

Appendix

A Generalizability of Distillation

Distribution Loyalty: We now assess the generalization of the KD process: whether the teacher and student behave similarly on unseen data. From the test set in Table 1, we take the instances that are not present in either the teacher or distillation data, resulting in 806k instances. We measure the Kullback-Leibler (KL) divergence between the output probabilities of the teacher and student models (trained with varying number of instances). KL values closer to zero, reflect similar distributions and behavior of the two models on the unseen data.

Figure 3 shows that with increasing amount of distillation data, the teacher and student models output highly similar probabilities. Further, we note that in some cases, such as for `hi-IN`, with 1% of the distillation data (or 6472 instances), the two models output highly diverging probabilities. Increasing the distillation data to 30% and upwards, we see that the probability distributions become highly similar, a fact also reflected in Table 3, where the student models have very close scores.

NCQ/CQ Class Separation: Figure 4 shows the class separation for the different student models, distilled with varying amount of data (1%, 10%, and 50%), and as well as the teacher model (representing the target performance for S). An ideal classifier would output 0 probability for NCQ class, and 1.0 for the CQ class, given that the IC models are trained on the binary case to predict whether an utterance is a commercial question or not.

From Figure 4, it can be noted that for $S_{1\%}$, there is a large amount of CQ utterances that have low CQ probability (x-axis). As the amount of distillation data is increased, we note that the distribution become more skewed (Dean and Illowsky, 2018), which represents an increase in classifier accuracy. For instance, from a skewedness score of $G1_{CQ} = -2.29$ for $S_{1\%}$, we obtain a skewedness score of $G1_{CQ} = -3.71$ for $S_{50\%}$, which implies that the CQ probability distribution is more skewed towards the higher CQ scores. In the case of classification models, the more skewed the distributions, the higher the classification performance.

90	0.021	0.028	0.028	0.025	0.031	0.036	0.023	0.064	0.032	0.039	0.059
80	0.012	0.017	0.018	0.014	0.023	0.018	0.009	0.052	0.021	0.022	0.048
70	0.022	0.03	0.032	0.017	0.036	0.035	0.035	0.067	0.034	0.037	0.061
60	0.015	0.02	0.02	0.018	0.024	0.026	0.014	0.052	0.026	0.036	0.054
50	0.013	0.019	0.025	0.011	0.026	0.025	0.013	0.06	0.023	0.035	0.059
40	0.014	0.023	0.017	0.014	0.021	0.016	0.022	0.059	0.024	0.038	0.052
30	0.021	0.03	0.021	0.01	0.028	0.031	0.024	0.073	0.031	0.035	0.062
20	0.036	0.045	0.045	0.049	0.044	0.055	0.034	0.118	0.059	0.063	0.112
10	0.034	0.048	0.054	0.042	0.05	0.049	0.036	0.132	0.057	0.068	0.134
5	0.071	0.087	0.094	0.064	0.063	0.083	0.1	0.177	0.09	0.141	0.191
1	0.134	0.145	0.203	0.154	0.156	0.202	0.216	0.351	0.223	0.256	0.58
	en-US	en-GB	es-ES	es-US	es-MX	fr-FR	fr-CA	de-DE	it-IT	pt-BR	hi-IN

Figure 3: KL-Divergence of the confidence score distribution between teacher and student models on unseen data. With increasing amounts of distillation data, the probability distributions become highly similar.

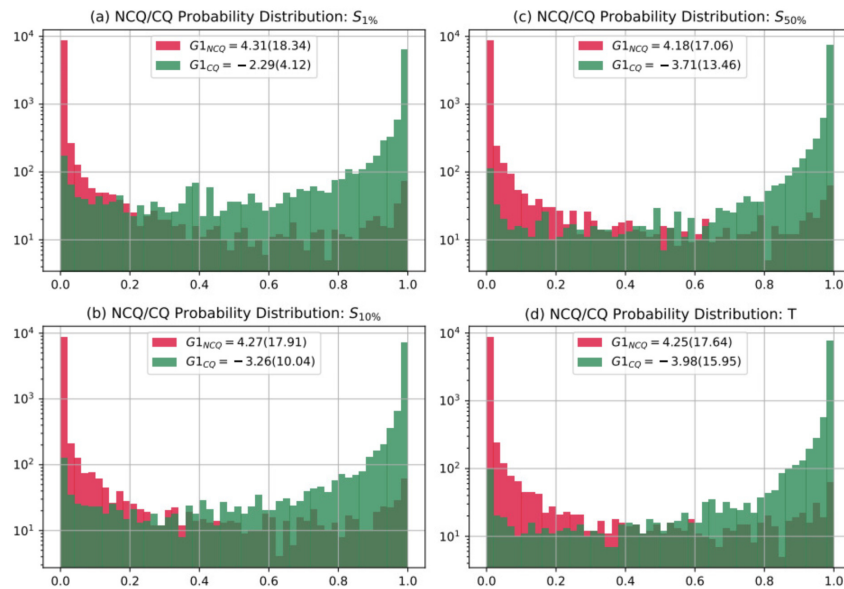


Figure 4: NCQ/CQ confidence distribution (x -axis) as a function of how likely an utterance is to be CQ. For NCQ, this probability ideally should be close to zero, and vice-versa for CQ (close to one). The skewedness score $G1$ measures the concentration of the probability mass for NCQ and CQ, respectively. For NCQ the higher score the better (in the positive range), whereas for CQ, the lower the score the better (negative range). The results are shown for the student models: $S_{1\%}$, $S_{10\%}$, $S_{50\%}$ (distilled with 1%, 10%, and 50% of the data, respectively), and for the teacher model T .

Bringing the State-of-the-Art to Customers: A Neural Agent Assistant Framework for Customer Service Support

Stephen Obadinma^b, Faiza Khan Khattak^a, Shirley Wang^{a,c}, Tania Sidhom^{a,b},
Elaine Lau^{d,g}, Sean Robertson^{a,c}, Jingcheng Niu^{a,c}, Winnie Au^a, Alif Munim^a,
Karthik Raja K. Bhaskar^e, Bencheng Wei^e, Iris Ren^e, Waqar Muhammad^e, Erin Li^e,
Bukola Ishola^e, Michael Wang^f, Griffin Tanner^f, Yu-Jia Shiah^g, Sean X. Zhang^g,
Kwesi P. Apponsah^g, Kanishk Patel^h, Jaswinder Narain^g, Deval Pandya^a,
Xiaodan Zhu^{a,b}, Frank Rudzicz^{a,c}, Elham Dolatabadi^{a,c}

^aVector Institute for Artificial Intelligence, ^bQueen’s University, ^cUniversity of Toronto,
^dMcGill University, ^eCIBC, ^fKPMG, ^gPwC Canada, ^hUniversity of Alberta, ^gMila

Abstract

Building Agent Assistants that can help improve customer service support requires inputs from industry users and their customers, as well as knowledge about state-of-the-art Natural Language Processing (NLP) technology. We combine expertise from academia and industry to bridge the gap and build task/domain-specific Neural Agent Assistants (NAA) with three high-level components for: (1) Intent Identification, (2) Context Retrieval, and (3) Response Generation. In this paper, we outline the pipeline of the NAA’s core system and also present three case studies in which three industry partners successfully adapt the framework to find solutions to their unique challenges. Our findings suggest that a collaborative process is instrumental in spurring the development of emerging NLP models for Conversational AI tasks in industry. The full reference implementation code and results are available at <https://github.com/VectorInstitute/NAA>

1 Introduction

Rising demand for AI-powered conversational agents (Fu et al., 2022; Sundar and Heck, 2022), especially for customer support service, is estimated to grow at a compound annual growth rate (CAGR) of 23.4%, earning a predicted revenue of around \$29.9B USD by 2028 (Market, 2018). As such, conversational AI research has increased substantially, especially to enhance customer service support (Nicolescu and Tudorache, 2022). Despite a proliferation of agent assistants from Microsoft, IBM, Oracle, and Google, there remain many unanswered questions and challenges (Fu et al., 2022) that need to be addressed before the widespread proliferation of AI in practice. We argue that bridging the gap between natural language processing (NLP) research in academia and industry is an overlooked issue in conversational AI.

Apart from the handful of aforementioned conglomerates, it is difficult for most other companies, which are not in the process of conducting cutting-edge NLP research, to benefit from the recent progress of NLP. Conventional conversational AI architectures are delicate and complex, and require a large degree of specialised knowledge to bring a full system into fruition. However, with the introduction of the large-scale pre-trained transformer-based language models such as BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019), it is possible for smaller teams to take advantage of this “monopoly” by being able to fine-tune these powerful models on their comparatively small amount of data and achieve high performance, therefore harnessing the architectural achievements of those powerful language models to devise their own conversational AI systems that exploit their more fine-grained expertise and knowledge of their customers’ needs.

The interdisciplinary nature of AI-enabled customer service support makes for an inherently difficult task (Nicolescu and Tudorache, 2022). To arrive at an answer to a user query, a system must bore through several layers of complexity: first, the intent behind the question must be categorized and quantized into a form which can be manipulated by the system; second, determining from thousands or millions of stored records the information relevant to that intent; and finally to manipulate said information into a form which the user may understand.

To handle such complexity, techniques from across NLP must be employed, raising the barrier-of-entry for companies with significant customer support needs. Larger, well-established companies can offer solutions as a service, but said solutions are often closed-source, trained on generic data with few in-domain terms, and may not be easily integrated into a company’s workflow.

This paper represents the cumulative efforts of

both researchers and industry practitioners over a yearlong project (whose details may be found in Appendix A) to develop state-of-the-art customer service support systems. In its publication, we hope to help lower the barrier-of-entry for future industry practitioners by inspiring similar collaborations. Our contributions are twofold and in line with the strengths of the contributors:

1. We release an open-source Neural Agent Assistant (NAA) based on state-of-the-art neural architectures (Fu et al., 2022). The implementations are well-documented, illustrating how the systems can be extended for specific use cases.
2. To wit, we explore three case studies in which three industry partners successfully adapt the agent to find solutions to their unique challenges.

In their presentation, we stress that a perfect, universal customer service system is unattainable as their real-world applications are not identical. As such, any dialogue about real-world NAA must feature a discussion of both the technologies shared with, and the differences between tasks.

2 Neural Agent Assistant framework for Customer Service Support

As the basis of our collaborative project, we established a common Neural Agent Assistant (NAA) framework to serve as a guide for industry partners to adapt to their companies’ workflows. Those technologies were made available in a clear, easy-to-use engineering pipeline, alleviating one of the greater challenges for industry in obtaining a foothold in building competent neural agents.

The framework features three high-level components shown in Figure 1: (1) Intent Identification, (2) Context Retrieval, and (3) Response Generation. Though designed as parts in a pipeline in generating a natural language response to customer questions (Figure 1 (b)), the output from any given component can nonetheless be transferred back to the human agent when needed (Figure 1 (a)).

2.1 Intent Identification

Intent identification is a critical component of NAA as it is often the first step in a customer service pipeline upon which all subsequent components depend on. Its goal is to classify the intent of

a given query. This may be binary, determining whether the query is relevant or irrelevant to the knowledge base (KB) (see Section 3). More often, the task involves classifying the query into a fine-grained, domain-specific intent class. For this purpose, we designed a query encoder to detect and understand customers’ input and classify it into one of N classes. The encoder consists of a pre-trained, Transformer-based language model (Vaswani et al., 2017), followed by a pooling layer, then a final linear layer. A query is fed as an input to the encoder, with the latter outputting a categorical probability distribution over the intents.

We chose BERT (Devlin et al., 2018) as the encoder, which we fine-tune and evaluate on two task-specific corpora: Banking77 (Casanueva et al., 2020) and CLINC150 (Larson et al., 2019). Following the methodology proposed by (Zhang et al., 2021), we first fine-tune the BERT model for the intent classification task using the Banking77 dataset, which is composed of 13k queries labelled into 77 intents. We then evaluate it on the CLINC150 dataset (banking domain) in the few-shot setting using all 15 classes. We achieved 92% F1-score for fine-tuning and 85% and 90% F1-score for one-shot and five-shot learning, respectively.

For NAA’s core system demonstration, the intent label is not used to guide the context retrieval and response generation, but it can be easily leveraged for knowledge extraction and abstractive summarization as shown in Section 3.

It is important to note that intent identification is only one of the main natural language understanding (NLU) components that a complete customer support agent would possess. Named-entity recognition (NER) is often an equally important task, where the goal is to identify any named entities, including relevant people, organizations, or products the user is inquiring about, as well as details about time and location, like for scheduling a restaurant reservation. There can be a great need for an agent to identify domain-specific entities in user dialog for the purposes of slot filling, hence there is a similar need for NER frameworks for easily adapting to domain-specific industry needs. For the sake of simplicity, and because an NER component was not requested from our industry partners, it is not included within the scope of the project. We leave developing this NER framework to future work.

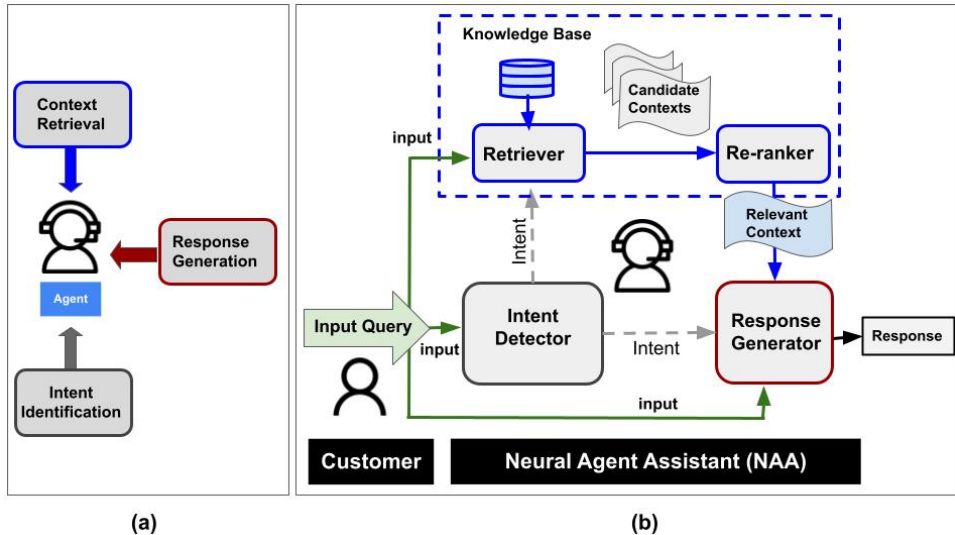


Figure 1: Neural Agent Assistant framework. (a) shows how a human agent is able to leverage the three NAA components for support. (b) shows the complete NAA pipeline, starting with an input query from a customer which is first fed to the NAA’s intent identification module. The intent label is generated and can be used to guide the context retrieval and response generator. The retriever and re-ranker work together to find the most relevant context candidate passages from the KB, which the response generator employs to craft a response to the user query.

Table 1: Primary results for Passage Retriever (PR) and the Re-Ranker (RR), comparing the performance of pre-trained and fine-tuned models with a variety of standard passage retrieval evaluation metrics (note higher results are better).

	MRR	MAP
PR pretrained		
msmarco-distilbert-base-tas-b	0.835	0.801
fine-tuned		
multi-qa-mpnet-base-dot-v1	0.886	0.860
msmarco-distilbert-base-tas-b	0.864	0.834
RR pretrained		
msmarco-MiniLM-L-6-v2	0.998	0.997
fine-tuned		
msmarco-MiniLM-L-6-v2	1.000	1.000

2.2 Context Retrieval

This component is designed to retrieve the most semantically similar documents from a predefined knowledge base (corpus of documents) given a user query. Documents or passages extracted from the KB are presumed to contain an appropriate answer to the query, though not in a concise form. Context retrieval is a two-step process starting with passage retrieval and followed by re-ranking. The retriever uses a bi-encoder design (Karpukhin et al., 2020), extracting supporting context from a knowledge-base at the level of paragraphs (hence asymmetric semantic search) based on SentenceBERT (Reimers and Gurevych, 2019).

The re-ranker is a BERT-based encoder designed for sequence-pair classification that scores the relevancy of all top-k retrieved candidate contexts for a given input query (Nogueira and Cho, 2019).

Our implementation includes pipelines to run the retriever and re-ranker pre-trained on MSMARCO (Bajaj et al., 2016) as-is, or to fine-tune both models using Multiple Negatives Ranking Loss on a knowledge-grounded, question-answering corpus such as ELI5 (Fan et al., 2019). The MSMARCO dataset was generated by sampling and anonymizing Bing usage logs. The dataset includes over 1 million queries with at least one human-generated answer per query, as well as relevant Wikipedia passages retrieved by Bing for each query. The ELI5 dataset is made up of complex questions and long and explanatory answers from Reddit users about random topics. Each question and answer pair is grounded in relevant Wikipedia passages for supporting information. Table 1 compares and reports the performance of the pretrained and fine-tuned models on the ELI5 dataset, illustrating the benefits of fine-tuning across systems and conditions.

Different alternative approaches have been taken in previous works for context retrieval, including combining dense passage retrieval with lexical rule-based retrieval such as BM25 to improve results and computational efficiency (Gao et al., 2021; Khazaeli et al., 2021), and using extensive pretrain-

Table 2: Primary results for the Response Generation component of NAA. Models were fine-tuned on MSMARCO.

Model	Test-set	F1-score	BLEU-1	Rouge-L
GPT2-medium	MSMARCO	40.2%	32.0%	36.0%
GPT2-large	MSMARCO	32.0%	20.4%	28.0%
DialoGPT-medium	MSMARCO	28.2%	15.3%	24.6%
GPT2-medium	ELI5	6.98%	0.1%	5.22%

ing on the encoders (Khattab and Zaharia, 2020). Although these methods have been shown to be effective in some cases, it adds further complexity and makes for a less generalizable approach compared to the BERT-based approach we take.

2.3 Response Generation

The ultimate goal of our NAA is to be able to automatically generate a human-like answer to a customer’s query. Therefore, the final component is dedicated to producing in-context natural-language responses to customers’ queries via a generative transformer model. The model is fed the ranked passages from the retrieval component and outputs the natural-language response. Though the input is extractive, the output is abstractive.

Our reference implementation provides pipelines for training GPT-2 models of any size (Radford et al., 2019) for the question answering (QA) task on both of the above-mentioned knowledge-grounded datasets: MSMARCO and ELI5. We used a multi-task loss combining language modeling with a next-sentence prediction objective. Following training, nucleus top-p sampling was used for decoding and text generation. The results for GPT-2 medium are presented in Table 2.

The three components of our pipeline - intent classification, context retrieval, and response generation - perform well on the domains and tasks they were trained on. However, as discussed in Section 1, a general system will fail to account for the specific challenges companies face related to the content and quality of NAA. As was illustrated, the pipeline above provides numerous opportunities for tailoring each function, opportunities leveraged by our industry partners in the next section.

3 Industry-specific Proof Of Concept Implementations

In this section, we demonstrate how the NAA framework outlined in Section 2 was adapted to

our industry partners’ use-cases and settings, which is continuing to be refined for deployment in real-world scenarios.

Canadian Imperial Bank of Commerce (CIBC) is a financial institution with an Advanced Analytics team focusing on finding the correct curated response to banking queries. KPMG is a professional services firm which is looking to NAA to service internal queries over large bodies of legal and regulatory documents. PricewaterhouseCoopers (PwC) is also a professional services firm looking to better advise its banking clients on building NAA technologies. In the following subsections we clarify the companies’ motivations in participating in the project, how they adapted our NAA framework, and their plans for deployment.

3.1 Company I: CIBC - NAA Tools for Banking Customer Service Support

The primary motivation for implementing and deploying NAA by CIBC is to support the bank’s continued focus on leveraging digital technologies to make clients’ banking experiences even better.

The pipeline modified by CIBC (see Figure 2 in Appendix A) consisted of four components. The first component was a binary classifier (i.e., a BERT encoder) which was trained to classify whether an input query has a banking intent. The inputs identified as being banking related were further processed by the assistant. Module 2 was the same as NAA’s intent identification component, though it was improved through data augmentation: the component was fine-tuned on back translation (Sennrich et al., 2016) (English → German → English) and insertion data. With data augmentation, F1-scores improved from 91.8% to 92.7%. Module 3 was a knowledge-driven QA system including both the Context Retrieval and Response Generation components of the NAA. If the banking intent from the previous layer had a confidence score greater than parity, then the top 5 most relevant financial contexts from a curated financial KB were retrieved and ranked. The KB included long-format questions and answers, as well as FAQs downloaded from the company’s publicly available website. The context, therefore, was question/answer pairs that, after being retrieved, provided a similar question along with an answer to the customer query. In addition, the GPT-2 Medium model pre-trained on MSMARCO was also used to gener-

Table 3: KPMG: Legal data NAA pipeline results. Passage size (PS) refers to number of sentences used for the passages (2, 3, and 4 for *short*, *medium*, and *long* respectively). Corpus refers to whether the dataset was preprocessed. Retriever refers to corpus embedding bi-encoder (all retrieval components are pretrained on MSMARCO). Response generator (RG) shows the pre-training dataset used for GPT-2-medium.

PS	Corpus	Retriever	RG	Manual Score	BLEU-1	F1-score
Med.	Raw	DISTILBERT	ELI5	44	0.193	0.218
Med.	Clean	DISTILBERT	ELI5	40	0.176	0.198
Short	Clean	DISTILBERT	ELI5	30	0.181	0.206
Long	Clean	DISTILBERT	ELI5	43	0.212	0.232
Med.	Clean	BERT	ELI5	49	0.303	0.336
Med.	Raw	BERT	MSMARCO	74	0.450	0.506
Med.	Clean	BERT	MSMARCO	N/A	0.459	0.443
Short	Clean	BERT	MSMARCO	N/A	0.422	0.426

ate human-like responses for the questions based on the retrieved long answers. Low-confidence queries were passed into the final component of the pipeline, which is based on KeyBERT, an architecture utilizing BERT, and deals with out-of-domain intent identification. Using the KeyBERT, keywords from the query were extracted and kept as new out-of-domain intents. Moreover, these new out-of-domain intents were recorded and tracked to enhance banking intent classification in the future by creating more relevant intent categories.

Following a successful implementation of the pipeline, a web application was developed to be used directly by clients. React and REST API service were used as the basis for the web application, which were integrated with the Amazon Web Services (AWS) cloud platform. Extensive developer testing has been conducted, and additional deployment include carrying out agent volunteer testing, which involves using several agents at the bank testing the application for optimal functionality and usability. Lastly, in order to make the system able to dynamically improve in response to customer feedback, the developers plan to implement a feedback loop using customer data. The feedback loop is a mechanism that helps determine how well the model works in production, and provides the necessary feedback to determine whether any changes are needed as a response to customers' user experience with the application and any desired improvements (see Figure 3 in the Appendix).

3.2 Company II: KPMG - A Q&A Tool for Legal Documents Analysis

The motivation for adapting NAA and its implementation by KPMG was reduce the inefficiencies

in providing accurate information to user queries in the legal domain drawn from agreements and reports. The company aims to improve their existing text search parsing relying on manual/keyword search using a more robust and easy-to-use novel knowledge-based document query tool based on a NAA.

The pipeline for this implementation included using and externally evaluating the context retrieval and response generation components of NAA framework on a curated legal dataset, CUAD v1 (Hendrycks et al., 2021), which contains over 13k sentences based on commercial legal contracts labelled into 41 types of legal clauses. A group of 5 experts including 3 data scientists and 2 legal domain data engineers helped create the legal domain knowledge grounded QA dataset and provided manual evaluation of the performance of the model on the response generation task. The legal QA dataset includes in-house human generated question and answer pairs (n=47) drawn from CUAD v1. In particular, the dataset is a collection of question-answer-evidence triplets, and a KB where the questions and answers were generated by the experts and were accompanied by a supporting context from the KB. The quality and accuracy of the generated responses by the pipeline was evaluated manually by one of the data engineers with legal domain expertise, as well as automatically by using BLEU and F1-scores as can be seen in Table 3. Through manual evaluation, each generated response was scored from 0 to 100 based on their quality and relevance, and the average score of all of them was calculated. The retriever and GPT-2-medium fine-tuned on MSMARCO achieved the highest score of 74 through manual evaluation, as well as a BLEU-1 and F1-score of 0.45 and 0.5, respectively.

A two-phase production deployment plan is implemented for the developed prototype. In phase I, the prototype is planned to be made available to internal employees only as a document processing tool. The service will be based on a cloud-hosted managed service environment, and would allow users to upload a set of documents and enter queries. Based on the queries, related context will be extracted from the documents and abstractive responses, which can be used in the formation of summary reports. Built on phase I, in phase II the tool will be made available to external clients and

integrated with their systems through an API for the their managed services platform.

3.3 Company III: PwC Canada - Transferring Emerging Technologies to Financial Customer Queries

The main motivation for adapting and applying NAA by PwC was to gain experience in creating an end-to-end NAA pipeline to process customer queries regarding bank account opening and mortgage applications. The aim was to later use the acquired knowledge to support small to mid-sized financial clients in building NAA chatbots that can reduce inefficiencies when dealing with large volumes of requests (particularly during high-volume seasons like home-buying seasons).

The first component of the prototype included a binary classifier to classify the text into either a “general” or “agent-related” topic to mitigate the answer generation model from confusing the user by responding to irrelevant queries. The binary classifier was trained on a labeled dataset including a total of 15,732 queries (10,331 “agent-related” and 5,401 “general”), and achieves an F1-score of greater than 0.95. The “general” queries were extracted from English-Second-Language practice conversations and common greetings, while “agent-related” queries were pulled from the Banking77 dataset (Casanueva et al., 2020). In case of a general query, a pretrained dialogue generation model, DialoGPT (Zhang et al., 2020), was triggered to make a basic response to the user.

The answer generation component included both the retrieval and response generation tasks. For the retrieval task, both the retriever and re-ranker provided by NAA (pretrained on MSMARCO) were used to extract relevant question-answer pairs from the KB. The KB included 250 long format questions and their respective answers downloaded from FAQs owned by top banks in Canada. Following the retrieval task, the top 3 contexts (including question-answer pairs similar to section 3.1) along with the user’s query were provided as inputs to the answer generation model which uses a GPT-2 model further fine-tuned on the 250 question answer sets. The generated answers were compared and evaluated on a set of new human-generated question answer pairs (n=30), achieving a Rouge1 F1-score of 0.088 and a RougeL of 0.084.

The deployment of this solution at the produc-

tion level is to combine NAA components with Automated Speech Recognition (ASR) tools using a backend service that determines whether a client request is audio or text, after which the request would be processed by the appropriate pipeline. An asynchronous web framework such as NodeJS’ Express or Python’s FastAPI is planned to be used to create a system scalable to multiple concurrent users, and using an Apache MXNet framework in combination with distributing the workload across multiple GPUs will support inference on the ASR and the response generation models.

4 Summary and Discussion

We summarize our experiences in this section.

Domain/task-specific system works better: As stated earlier, the task/domain agnostic system is not well-suited for different use-cases as our industry partners found, and as confirmed by our experiences. Real-world conversational datasets are noisy and contain domain-specific concepts, and as a result, domain agnostic systems that are trained on clean open-source datasets are likely to have poor performance when evaluated on realistic inputs. The best practice would be to have a system designed/adapted based on domain and task.

Finding a good performance evaluation method: Due to the inherent difficulty in evaluating the quality of retrieved contexts and generated answers, finding a suitable model assessment strategy is imperative (Khapra and Sai, 2021), which includes the processes of model selection, bench-marking, and deciding between manual and automated assessment. Automatic evaluation metrics in particular, although flawed, allow for the ability to benchmark many different model variations, leading to a strong model being picked if there is a general agreement between different metrics. The companies found that using a combination of automatic evaluation metrics like BLEU score, F1-score, Rouge-1, along with expert manual evaluation, helped better inform model selection.

Efficiency and optimization: The efficiency of the algorithm/model being used is important, as is weighing trade-offs and discovering optimizations. For example it was noted that BERT-base models tend to perform better than DistilBERT-base for the bi-encoder, but at the cost of longer inference time. Hence, it is important to consider multiple different

approaches when implementing a component and deciding what is the most important aspect of performance. Another challenge that was identified by one company was designing an efficient algorithm for parsing customer inputs into meaningful entities, which was solved by using a custom built parser tailored specifically for in-domain data.

Effectively Combining Academia and Industry Expertise:

This collaborative project combined the expertise from academia and industry, which resulted in successful implementation of systems to be deployed at the industry participants' companies. We tried to keep the whole process smooth by having regular meetings and adding documentation. This helped us to collaborate in an efficient and effective manner while keeping engagement throughout the whole process.

5 Conclusion

In this paper, we present a collaborative project for building three customized NAA systems for different tasks and domains in industry. Through our framework and collaborative process, we have tried to bridge the gap between industry and cutting edge NAA technologies by providing much needed open-source tools and expertise in foundational language models and conversational AI. We have observed this framework being successfully adapted into real-world use-cases to enhance customer service support, and it has helped spur NAA development in industry. We hope that our current effort will serve as a valuable use-case to the community, and it is our plan to continue similar collaborative efforts in future in other areas of AI.

6 Ethical Considerations

A limitation of the natural language response generation module of our framework is that there are few guarantees that the model does not produce erroneous or harmful responses in response to user queries. As the base of this module is a language model such as GPT-2 that is pretrained on a massive corpus of data that has not been carefully ensured to be debiased and free from harmful or prejudiced content, it can be susceptible to adversarial inputs and thus has the potential to produce bigoted responses even to benign user queries (Kirk et al., 2021). This presents ethical concerns when having the NAA deal directly with customers, and therefore, to avoid such concerns, we recommend

that the NAA in its current form only be used in a human-in-the-loop process, where a human agent works closely with the NAA for assistance. The human agent can ensure that any potential customer or client does not receive any unwanted responses from the NAA.

Acknowledgements

We would like to acknowledge and give thanks to Sedef Akinli Kocak and Gerald Penn from the Vector Institute for Artificial Intelligence for their help in facilitating this project. We also acknowledge the contributions of Andrew Brown from CIBC on CIBC's NAA implementation.

References

- Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.
- Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. *Efficient Intent Detection with Dual Sentence Encoders*. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at <https://github.com/PolyAI-LDN/task-specific-datasets>.
- 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.
- Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. *ELI5: Long Form Question Answering*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.
- Tingchen Fu, Shen Gao, Xueliang Zhao, Ji-rong Wen, and Rui Yan. 2022. Learning towards conversational AI: A survey. *AI Open*, 3:14–28.
- Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. *COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3030–3042, Online. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. CUAD: An Expert-Annotated NLP Dataset for Legal Contract Review. *NeurIPS*.

- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense Passage Retrieval for Open-Domain Question Answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Mitesh M Khapra and Ananya B Sai. 2021. A tutorial on evaluation metrics used in natural language generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorials*, pages 15–19.
- O. Khattab and Matei A. Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Soha Khazaeli, Janardhana Punuru, Chad Morris, Sanjay Sharma, Bert Staub, Michael Cole, Sunny Chiu-Webster, and Dhruv Sakalley. 2021. [A Free Format Legal Question Answering System](#). In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 107–113, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Hannah Rose Kirk, Yennie Jun, Filippo Volpin, Haider Iqbal, Elias Benussi, Frederic Dreyer, Aleksandar Shtedritski, and Yuki Asano. 2021. Bias out-of-the-box: An empirical analysis of intersectional occupational biases in popular generative language models. *Advances in neural information processing systems*, 34:2611–2624.
- Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. [An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction](#). 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)*.
- Gold Nanoparticles Market. 2018. Global Industry Analysis, Size, Share, Growth, Trends, and Forecast 2017–2026. *Transparency Market Research*.
- Luminița Nicolescu and Monica Teodora Tudorache. 2022. Human-Computer Interaction in Customer Service: The Experience with AI Chatbots—A Systematic Literature Review. *Electronics*, 11(10):1579.
- Rodrigo Nogueira and Kyunghyun Cho. 2019. [Passage Re-ranking with BERT](#). *arXiv e-prints*, page arXiv:1901.04085.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.
- 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*. Association for Computational Linguistics.
- 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.
- Anirudh Sundar and Larry Heck. 2022. Multimodal Conversational AI: A Survey of Datasets and Approaches. *arXiv preprint arXiv:2205.06907*.
- 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.
- 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](#).
- Haode Zhang, Yuwei Zhang, Li-Ming Zhan, Jiaxin Chen, Guangyuan Shi, Xiao-Ming Wu, and Albert Y. S. Lam. 2021. Effectiveness of Pre-training for Few-shot Intent Classification. *arXiv preprint arXiv:2109.05782*.
- Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020. DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In *ACL, system demonstration*.

A Appendix

Project participants: The participants of this project consist of:

- *Technical and project management teams from Vector Institute for AI* initiating this project with the mission to bring industry and academia closer so that the two groups could learn and benefit from each other.
- *Data scientists from three industry partners* participated to enhance their knowledge and apply new methods in their companies.
- *University faculty from partnering universities* providing additional advising and exposure to the state-of-the-art methods.

Project steps: The following are the points describing the project steps and participation.

- Vector Institute for AI:
 - defined the problem statement and project scope with the help of some input from the industry partners,
 - implemented general domain-agnostic NAA reference implementations consisting of three high-level components: (1) Intent Identification, (2) Context Retrieval, and (3) Response Generation,
 - provided tutorial, open-source datasets, computing services, training, and constant feedback & support.
- Faculty provided advising on cutting edge technologies.
- Industry partners (CIBC, KPMG, PwC):
 - identified the use-cases,
 - modified, trained/re-trained and improved the framework on the datasets of their choice according to their use-cases and industry settings.
 - developed their own domain/industry specific end-to-end pipeline to deploy in their respective companies.

B Environment Setup

The following resources were provided by the *Vector Institute for AI* to the participants.

- **Git repos:** Git repositories containing full reference implementation code and training details were provided by the *Vector Institute for AI*¹.
- **Cluster and GPU access:** Access to cluster and GPUs was provided by the *Vector Institute for AI*. Model training was conducted on a remote cluster on one of 3 GPUs: NVIDIA T4 (16GB), NVIDIA® Tesla P100 (16GB), and NVIDIA RTX 6000 GPU (24GB)
- **Dataset storage:** All datasets used were made available on cluster. Moreover, each participant had 50GB storage space (which could be increased upon request) to store datasets, model checkpoints, and other files necessary to formulate their solutions.
- **Google Cloud Platform (GCP) access:** The participants were provided access to GCP and TPUs. Also a GCP training session from Google was hosted by the *Vector Institute for AI*. The purpose of this platform was to provide the participants with experience using cloud services to train and deploy their system.

C Reference Implementation Training Details

In this section, we provide some of the pre-processing decisions, training details, and hyperparameters for our reference implementations for replication purposes. For our implementation of the transformer-based language models we use the python libraries, Huggingface Transformers (Wolf et al., 2019) for BERT and GPT-2, and SentenceTransformers (SBERT) (Reimers and Gurevych, 2019) for the bi-encoder and cross-encoder. We use Pytorch as the backbone of our implementations. An AdamW optimizer is used for the training of all of the models.

- **Intent Identification:** The max sequence length for BERT is set to be 300 tokens. The model is fine-tuned for 40 epochs using a batch size of 16 and a learning rate of $2e-5$. We utilize a linear scheduler with a warmup ratio of 20%. We used 5% of the training set for model validation purposes, and the provided

¹<https://github.com/VectorInstitute/NAA>

Banking77 test set for evaluation. Early stopping based on validation loss with a patience of 2 was used to select the best performing model. The training settings for the few shot model include using a frozen BERT encoder previously fine-tuned on Banking77 as a feature extractor for few shot classification. A new linear layer on top of the BERT encoder is trained to classify the new 15 classes from CLINC150. The linear layer is trained for 5 epochs with a batch size of 6 and a learning rate of 0.001. The rest of the training settings are similar to before.

use the existing train and validation sets for fine-tuning and evaluating the model.

- **Context Retrieval:** For the version of the bi-encoder fine-tuned on the ELI5 dataset, the dataset is preprocessed such that a re-ranker pretrained on MSMARCO re-ranks the original 7 wikipedia passages per query, and picks the most relevant passage to be part of the question-answer training pair. The base *msmarco-distilbert-base-tas-b* bi-encoder is fine-tuned on this dataset for 3 epochs with a batch size of 16. A scheduler with a warmup ratio of 10% is also used. 15% of the training data is set aside for testing. The cross encoder has the same hyperparameters, with the addition of the max sequence length being set to 512 tokens. A learning rate of $2e-5$ is used for fine-tuning both modules.
- **Answer Generation:** Lastly, the GPT-2 answer generation model generates an output sequence that is a maximum of 200 tokens, at a temperature of 0.7, and with `top_k` and `top_p` values of 100 and 0 respectively. The model is trained for 5 epochs on MSMARCO using a batch size of 1 (with 8 gradient accumulation steps simulating a batch size of 8), and a learning rate of $5e-5$. Furthermore, the maximum number of input tokens is set to 330. Additional settings include setting the language modelling loss coefficient to 10.0, applying gradient clipping with a magnitude of 10.0, and using a multiple choice classification head for the second head of GPT-2 with a loss coefficient of 1.0. The MSMARCO dataset is preprocessed such that we use the well-formed answer if possible, and avoiding using questions for which there are multiple answers without a well formed answer. We

D Supplementary Figures

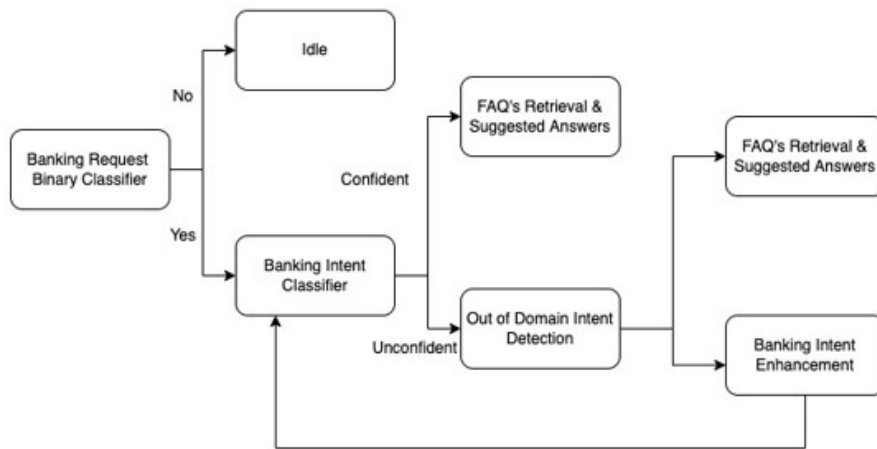


Figure 2: CIBC Banking Agent Assistance Implementation Pipeline.

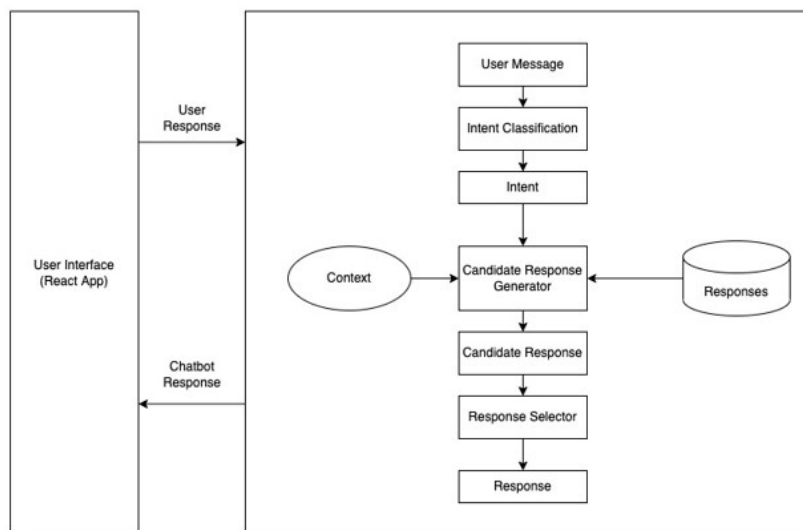


Figure 3: CIBC - Diagram showing the Banking Chatbot Service API. The user (human agent) interacts with the NAA pipeline through the React App, and sends a response. Then the NAA pipeline processes the request and a chatbot sends a response.

Zero-Shot Dynamic Quantization for Transformer Inference

Yousef El-Kurdi* Jerry Quinn* Avirup Sil

IBM Research AI

{yousefelk, jlquinn, avi}@us.ibm.com

Abstract

We introduce a novel run-time method for significantly reducing the accuracy loss associated with quantizing BERT-like models to 8-bit integers. Existing methods for quantizing models either modify the training procedure, or they require an additional calibration step to adjust parameters that also requires a selected held-out dataset. Our method permits taking advantage of quantization without the need for these adjustments. We present results on several NLP tasks demonstrating the usefulness of this technique.

1 Introduction

Transformer-based Neural Networks (NN) such as BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLM-R (Conneau et al., 2019), pre-trained on large amounts of data, have led to state-of-the-art (SOTA) results on many NLP tasks such as machine translation (Zhu et al., 2019), text classification (Wang et al., 2018) and question answering (Kwiatkowski et al., 2019; Clark et al., 2020). However, run-time inference of such large models is very costly due to their large computational requirements. In addition, deploying these models on smaller footprint mobile devices (Ravi and Kozareva, 2021) or cost-effective (Sanh et al., 2019; Jiao et al., 2020) CPU based machines require aggressive optimization techniques for both speed and network size. One popular speed optimization technique is NN quantization (Gholami et al., 2021; Kim et al., 2021; Zafrir et al., 2019), where network weights and activations are transformed from 32-bit floating-point representations to integers (typically 8-bit). Running inference using integer operations has two key advantages. First, the model size footprint is considerably reduced *e.g.* 8-bit quantization shrinks models by a factor of four. Second, inference throughput is significantly increased by using more efficient integer-based “single instruction

multiple data” (SIMD) (Hennessy and Patterson, 2012) instructions while improving memory bandwidth utilization, which is typically a bottleneck limiting computational throughput for NNs (Quinn and Ballesteros, 2018).

Fundamentally, quantization leads to a quantitative loss of information due to the lowered numerical precision. As a result, applying integer quantization directly to NN models leads to considerable drop in accuracy (Zafrir et al., 2019). However, by carefully adjusting the quantization parameters such as the clipping thresholds, the accuracy loss can be significantly reduced, if not eliminated.

The majority of quantization research (Gholami et al., 2021) involve a mix of quantization-aware training (QAT) and post-training calibration techniques with varying complexities to resolve the quantization performance gap. Several works (Kim et al., 2021; Choi et al., 2018; Zhou et al., 2017; Choi et al., 2018; Krishnamoorthi, 2018; Louizos et al., 2019; McKinstry et al., 2019) detail techniques for QAT as well as approaches where the quantization parameters are optimized using statistics gathered during training. While these approaches typically close the gap in the quantized model accuracy, they requires access to the training pipeline as well as the training data. In addition, these methods are not applicable to black-box models where both training procedures and data are not available. Also, these methods may be affected by training instabilities, increasing the complexity of the training regimes as described in (Krishnamoorthi, 2018). Post-training approaches such as (Migacz, 2017; Bhandare et al., 2019) require calibration techniques on selected datasets. For example, in (Migacz, 2017) KL-divergence (Kullback and Leibler, 1951) between the unquantized and quantized activations on each layer was used to tune the quantization clipping thresholds. Special care needs to be taken when selecting a calibration dataset; as it needs to be diverse enough but yet task

*Equal contribution.

specific. In certain cases this leads to low accuracy, or even unpredictable behaviour, if the run-time input deviates from the calibration dataset.

Two methods that share our high-level goals of eliminating the need for training datasets are introduced in (Nagel et al., 2019; Cai et al., 2020). These methods are implemented with CNN-based (Gehring et al., 2017) networks, and are used for image classification and object detection tasks. (Nagel et al., 2019) reduces the quantization error by re-scaling the weights of consecutive CNN layers while taking advantage of the equivariance property of the piece-wise linear ReLU function. (Cai et al., 2020), on the other hand, tunes the quantization parameters using synthetic data generated utilizing mean and variance statistics obtained from the batch normalization layers of the model itself. While both methods are applicable for mainly CNN-based networks, our algorithm is considerably simpler to implement and targets transformers (Vaswani et al., 2017); particularly SOTA NLP networks with BERT-like (Devlin et al., 2018; Liu et al., 2019) pre-trained representations.

In this work, we present a method that utilizes the Interquartile Range (IQR) (Tukey et al., 1977; Rousseeuw and Croux, 1993), which is a measure of statistical dispersion, to clip the activations dynamically during inference time. Our method ensures that at least 75% of the token-wise extreme activations are not modified, while leaving the remaining 25% to be statistically modified as outliers, leading to a robust behaviour while considerably improving quantization accuracy. Our method works for any transformer-based “trained” model and does not require any form of training or calibration. Overall, our contributions can be summarized as follows:

- We propose a novel “ready-to-use” inference-time dynamic quantization method that does not require sophisticated re-training/fine-tuning and additional calibration strategies.
- Empirically our proposed model demonstrates both effectiveness and robustness on several different NLP benchmark tasks.
- Further, contrary to prior work, experiments suggest that our proposed method works both for monolingual and multilingual transformer architectures out-of-the-box.

2 Methodology

2.1 Background

Existing approaches to speeding up inference for Transformers mostly focus on General Matrix Multiply (GEMM) operations. Fast GEMM implementations routinely use GPU and CPU specific SIMD instructions, to execute many multiplications and additions in parallel. They also optimize memory access patterns to make the best use of available memory bandwidth. Integer quantization speeds up the GEMM operations by increasing the amount of data transferred with each memory transaction. They also take advantage of denser SIMD instructions. For example, 8-bit quantization packs four times the data per memory transaction compared to 32-bit floating point values. Many CPUs also support 8-bit SIMD multiplication operations, providing faster as well as cost-effective computation.

2.1.1 Uniform Quantization

Dynamic quantization for inference quantizes activations at run time. The model weights are typically quantized once ahead of execution. Let $\mathcal{M} \in \mathbb{R}^{m \times n}$ be a matrix of either an activation or parameter weights. The quantization scale (QS) is obtained as:

$$QS = \max_{\substack{\forall i \in \{1, \dots, m\} \\ \forall j \in \{1, \dots, n\}}} |\mathcal{M}(i, j)|. \quad (1)$$

The matrix \mathcal{M} is then quantized to $\bar{\mathcal{M}} \in \mathbb{Z}^{m \times n}$ as follows:

$$\bar{\mathcal{M}} = \text{int} \left(\frac{2^b/2 - 1}{QS} \mathcal{M} \right), \quad (2)$$

where b is the number of integerization bits, typically 8, and the function `int` is the element-wise integer conversion operator; e.g. a floor function. The reason for the subtraction by 1 in (2) is to ensure that the quantization range is equally spread around zero. In the case of 8 bits, the range becomes ± 127 . This formulation also results in a symmetric form of uniform quantization, where the quantization is evenly split around zero. This can be modified by adding a zero-shift resulting in an asymmetric quantization (Krishnamoorthi, 2018), which may particularly be useful for certain activation functions such as ReLU (Nair and Hinton, 2010) and GELU (Hendrycks and Gimpel, 2016). While non-uniform quantization (Gholami et al., 2021) has been explored to better capture weight and activation distribution with variable step sizes,

uniform quantization leads to more efficient implementation on current hardware such as GPUs and CPUs with acceptable accuracy. Once matrices are quantized, GEMM operations can be performed using integer arithmetic allowing the use of fast SIMD instruction sets.

Quantization lowers numerical precision which leads to loss of information. Examining (1) shows how the QS can increase precision errors if it takes extreme values that largely deviate from the majority activations. Therefore, the activation tensor must be clipped to reduce the quantization error; however, excessive clipping can lead to distortions in the activation which also leads to drops in accuracy.

In the following section, we will outline a method that chooses better QS values for each activation tensor dynamically during inference, without any modification to the training pipeline or any requirement for calibration procedures.

2.2 Interquartile Range Clipping

If we consider the extreme values in the activations as outliers in a distribution, there is a substantial amount of research for identifying outliers (Ben-Gal, 2005; Hodge and Austin, 2004). Our solution makes use of a low complexity univariate statistical-based method for outlier detection referred to as the Interquartile Range (IQR) method originally proposed by Tukey (Tukey et al., 1977).

IQR is also considered a robust statistical measure (Rousseeuw et al., 2011) of the data spread, with the notion of robustness being defined using the concept of a *breakdown point* (Rousseeuw and Croux, 1993; Rousseeuw et al., 2011). The breakdown point is the minimum number of data that can be arbitrarily replaced while keeping the statistical measure bounded. The sample mean and variance have a 0 breakdown point, meaning that these measures are changed by even a single outlier; on the other hand, the IQR has a 25% breakdown point, making it a stable measure even if up to 25% of the data are outliers.

We introduce an algorithm that effectively uses IQR to clip outliers from an activation tensor which consequently improves the selection of the quantization scale as in (1). It is worth noting that a direct implementation of the IQR method is too slow as it uses a sorting operation in order to identify the quartiles on the data. The complexity of a naive implementation would be $\mathcal{O}(N \log N)$ where N

is the number of elements of the activation tensor. In the case of BERT-like models, $N = L \times H$, where L is the sequence length and H is the hidden dimension; e.g. for BERT-Large, $N = 512 \times 1024$. To lower this complexity, we obtain the IQR clipping threshold from a reduced set formed by taking the maximums, in absolute sense, along the H dimension. We will refer to this algorithm as the Token-Maximums IQR (TM-IQR) clipping. The resulting complexity of the IQR clipping becomes $\mathcal{O}(N + L \log L)$. Our experiments show that adding this form of IQR clipping slows inference by less than 2%, which is negligible considering the resulting accuracy gains.

Algorithm 2 Activation clipping using TM-IQR

```

Input: Activation tensor  $\mathcal{A} \in \mathbb{R}^{L \times H}$ 
 $\mathcal{L} \leftarrow \{1, 2, \dots, L\}$ 
 $\mathcal{H} \leftarrow \{1, 2, \dots, H\}$ 
1:  $M(i) \leftarrow \max_{j \in \mathcal{H}} |\mathcal{A}(i, j)|, \forall i \in \mathcal{L}$ 
2:  $M \leftarrow \text{sort}(M)$ 
3:  $q1 \leftarrow \text{first-quartile}(M)$ 
4:  $q3 \leftarrow \text{third-quartile}(M)$ 
5:  $t \leftarrow q3 + 1.5(q3 - q1)$ 
6:  $\mathcal{A}(i, j) \leftarrow \min(\mathcal{A}(i, j), t), \forall (i, j) \in \mathcal{L} \times \mathcal{H}$ 
7:  $\mathcal{A}(i, j) \leftarrow \max(\mathcal{A}(i, j), -t), \forall (i, j) \in \mathcal{L} \times \mathcal{H}$ 
Return:  $\mathcal{A}$ 

```

Algorithm 2 outlines the basic procedure of our TM-IQR clipping. In Line 1 we compose the set of token-maximum activations in the absolute sense. Essentially, we are reducing the set of activations to a smaller representative set that contains the top outliers of the larger set. Lines 2 to 5 compute the IQR threshold t which is then used to clip the entire activation tensor in lines 6 and 7. The value 1.5 in line 5 is commonly referred to as the IQR scale. It was historically proposed by Tukey (Tukey et al., 1977) as a level to detect outliers. It is possible to attempt to fine-tune this value, however we chose to use the historical value without tuning in line with the objective of our paper.

It is important to note that the TM-IQR algorithm assigns a dynamic clip value for each activation tensor as opposed to using a fixed value for all run-time inference. Unlike fixed clipping tuned by training datasets, we expect TM-IQR clipping to be applied in a zero-shot approach across multiple tasks while maintaining reasonable empirical accuracy. This is due to the fact that our clipping strategy guarantees that at least 75% of the row-wise extreme activations are not impacted by it, while a fixed clipping method does not offer such guarantees for all types of input, as is the

CPU	Precision	Method	Batch	WPS
Xeon 8260	int8	none	48	29005
Xeon 8260	int8	IQR	48	28640
V100	fp16	none	128	71998

Table 1: IQR throughput cost in WPS (words per sec) averaged over 4 runs. Each input is 512 tokens. 48 core Xeon 8260 and V100 speed included for reference.

case when the input is not very aligned with training data. This has the important effect of limiting the distortion error, which occurs when quantizing activations with excessive clipping.

3 Experimental Setup

Our run-time inference engine, implemented in C++, supports both FP32 and optimized 8-bit integer quantized inference (I8). We quantize model weights at load-time and dynamically quantize activations at run-time. The TM-IQR technique is a straightforward modification with a negligible impact on inference speed, as shown in Table 1.

3.1 TM-IQR

TM-IQR can be applied to the activations before each quantized GEMM operation. However, we found that the second feed-forward GEMM, henceforth referred to as FF2, contributes the majority of the quantization error. The input dimension of FF2 is very wide, $4 \times H$, providing more of a chance for saturation and integer numerical instability to accumulate. In addition, the input to FF2 constitutes the activations of either ReLU or a GELU nonlinearities. The range of such activation functions is unbounded on the positive side, which further increases the chance of saturation. Therefore, we found it most effective to apply the TM-IQR to the input activations of the FF2 GEMM operation.

3.2 Tasks

We test our proposed method on GLUE (Wang et al., 2018) and 2 popular question answering (QA) tasks: Natural Questions (NQ) (Kwiatkowski et al., 2019) and TyDI¹ (Clark et al., 2020). We train all our tasks using the publicly available (Wolf et al., 2019). For GLUE tasks, we run 5 seeds with hyper-parameters using HuggingFace’s defaults for BERT while tuning the learning rate for RoBERTa (refer to A for more details). For QA tasks, we follow (Alberti et al., 2019; Clark et al., 2020). Our

¹Note that TyDI is multilingual among 11 typologically diverse languages.

Task	FP32	I8	TM-IQR
XLM-R-base TyDI	67.7	62.9	67.0
XLM-R-large TyDI	68.8	66.8	68.4
XLM-R-base NQ	54.6	48.0	53.4
XLM-R-large NQ	56.6	53.3	56.1

Table 2: Question Answering performance.

underlying pre-trained language model for GLUE is both BERT (cased) (Devlin et al., 2018) and RoBERTa (Liu et al., 2019), while for QA, we used XLM-R (Conneau et al., 2019). Note our method *does not need* any fine-tuning once this step is done and models are obtained.

4 Results

Since our method does not modify the training pipeline or tune the quantization parameters on training sets, we compare our results directly to the FP32 numbers. We are not expecting our method to outperform FP32 but rather to reduce the negative effect of quantization while keeping its speed as well as simplifying the model deployment process.

4.1 Question Answering

On TyDI and NQ (Table 2), TM-IQR clearly recovers most of the performance lost to dynamic quantization and is superior to I8 by 1 point on average. Similar to GLUE, TM-IQR still performs well with the I8 drop being the highest.

4.2 GLUE

Table 3 shows that TM-IQR is robust with an overall average score drop, compared to FP32, by *only* 0.2% for BERT-base, 0.5% for BERT-large, 1.2% for RoBERTa-base and 0.4% for RoBERTa-large. For all 4 pretrained models, TM-IQR wins on average. Even when TM-IQR does not outperform I8, the loss is relatively small. Interestingly, TM-IQR does well for cases where I8 drop is large, *e.g.* CoLA and RTE for all models and STS-B for RoBERTa-base.

5 Conclusion

We show that BERT-like models can be quantized to 8-bit integers with good accuracy without the need to modify training procedures or add extra data sets for parameter calibration. We present a robust statistically-based algorithm that dynamically adjusts the quantization clipping to maintain reasonable accuracy. Our empirical results demonstrate the effectiveness of our method on a number

Task	FP32	I8	TM-IQR
BERT-base-cased			
MNLI	83.7 (0.2)	82.3 (0.5)	83.5 (0.3)
MNLI-MM	84.1 (0.1)	82.9 (0.2)	83.8 (0.2)
CoLA	58.0 (1.4)	48.3 (0.9)	57.7 (1.6)
SST-2	92.3 (0.3)	92.1 (0.2)	92.0 (0.4)
MRPC	88.5 (1.2)	88.8 (1.6)	88.5 (1.5)
STS-B	88.3 (0.8)	87.7 (0.8)	88.1 (0.8)
QQP	87.4 (0.1)	86.2 (0.3)	87.2 (0.2)
QNLI	90.8 (0.2)	90.3 (0.1)	90.5 (0.2)
RTE	64.6 (1.0)	63.9 (1.0)	64.9 (1.6)
Average	82.0	80.3	81.8
BERT-large-cased			
MNLI	86.4 (0.1)	86.0 (0.2)	86.0 (0.1)
MNLI-MM	86.5 (0.2)	86.3 (0.1)	86.3 (0.2)
CoLA	62.9 (0.8)	60.6 (1.5)	62.1 (1.2)
SST-2	93.3 (0.5)	92.8 (0.7)	92.9 (0.4)
MRPC	90.5 (0.5)	89.6 (0.9)	90.5 (0.7)
STS-B	89.6 (0.6)	87.4 (1.2)	89.1 (0.3)
QQP	88.3 (0.2)	88.1 (0.1)	88.1 (0.1)
QNLI	92.4 (0.1)	91.9 (0.1)	92.2 (0.2)
RTE	69.8 (1.4)	64.0 (2.0)	68.5 (1.7)
Average	84.4	83.0	84.0
RoBERTa-base			
MNLI	87.0 (0.1)	85.8 (0.3)	86.1 (0.1)
MNLI-MM	87.1 (0.1)	85.8 (0.2)	86.1 (0.1)
CoLA	53.7 (1.9)	22.7 (4.7)	50.8 (1.8)
SST-2	93.9 (0.2)	93.4 (0.4)	93.5 (0.3)
MRPC	78.6 (2.7)	77.2 (1.1)	78.4 (2.0)
STS-B	87.1 (0.8)	69.6 (0.8)	85.5 (0.8)
QQP	88.3 (0.1)	87.2 (0.2)	87.6 (0.1)
QNLI	92.5 (0.1)	90.1 (1.9)	91.4 (0.3)
RTE	68.0 (2.1)	64.7 (2.5)	67.4 (2.8)
Average	82.0	75.2	80.8
RoBERTa-large			
MNLI	90.6 (0.0)	90.0 (0.2)	90.3 (0.1)
MNLI-MM	90.0 (0.3)	89.6 (0.1)	89.6 (0.2)
CoLA	63.5 (0.6)	63.1 (1.3)	63.4 (0.6)
SST-2	96.3 (0.4)	95.8 (0.2)	95.8 (0.4)
MRPC	89.6 (0.4)	90.1 (0.7)	88.7 (0.8)
STS-B	91.8 (0.1)	91.2 (0.2)	91.4 (0.2)
QQP	89.8 (0.1)	89.4 (0.2)	89.4 (0.2)
QNLI	94.6 (0.2)	94.1 (0.3)	94.1 (0.3)
RTE	77.7 (2.0)	76.0 (5.1)	77.3 (1.5)
Average	87.1	86.6	86.7

Table 3: The TM-IQR clipping algorithm on GLUE tasks with three computational modes, 32-bit floating-point (FP32), 8-bit quantization (I8) and our algorithm TM-IQR. Metric values are mean and standard deviation (in parenthesis) over 5 seeds.

of NLP monolingual and multilingual tasks, trained on both base and large size BERT-like models.

References

- Chris Alberti, Kenton Lee, and Michael Collins. 2019. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.
- Irad Ben-Gal. 2005. Outlier detection. In *Data mining and knowledge discovery handbook*, pages 131–146. Springer.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. *CoRR*, abs/1906.00532.

Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178.

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.

Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. **PACT: parameterized clipping activation for quantized neural networks**. *CoRR*, abs/1805.06085.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. **Tydi QA: A benchmark for information-seeking question answering in typologically diverse languages**. *CoRR*, abs/2003.05002.

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*.

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. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

William B. Dolan and Chris Brockett. 2005. **Automatically constructing a corpus of sentential paraphrases**. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252. PMLR.

Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*.

- Dan Hendrycks and Kevin Gimpel. 2016. [Bridging nonlinearities and stochastic regularizers with gaussian error linear units](#). *CoRR*, abs/1606.08415.
- John L. Hennessy and David A. Patterson. 2012. *Computer Architecture - A Quantitative Approach, 5th Edition*. Morgan Kaufmann.
- Victoria Hodge and Jim Austin. 2004. A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126.
- S. Iyer, N. and Dandekar, and K. Csernai. 2017. [First quora dataset release: Question pairs](#).
- 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.
- Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. 2021. I-bert: Integer-only bert quantization. *International Conference on Machine Learning*.
- Raghuraman Krishnamoorthi. 2018. [Quantizing deep convolutional networks for efficient inference: A whitepaper](#). *CoRR*, abs/1806.08342.
- S. Kullback and R. A. Leibler. 1951. [On information and sufficiency](#). *Annals of Mathematical Statistics*, 22(1):79–86.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.
- Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.
- 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*.
- Christos Louizos, Matthias Reisser, Tijmen Blankevoort, Efstratios Gavves, and Max Welling. 2019. [Relaxed quantization for discretized neural networks](#). In *International Conference on Learning Representations*.
- Jeffrey L McKinstry, Steven K Esser, Rathinakumar Appuswamy, Deepika Bablani, John V Arthur, Izzet B Yildiz, and Dharmendra S Modha. 2019. Discovering low-precision networks close to full-precision networks for efficient inference. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*, pages 6–9. IEEE.
- Szymon Migacz. 2017. Nvidia 8-bit inference with TensorRT. In *GPU Technology Conference*.
- Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. 2019. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334.
- Vinod Nair and Geoffrey E. Hinton. 2010. [Rectified linear units improve restricted boltzmann machines](#). In *ICML*, pages 807–814. Omnipress.
- Jerry Quinn and Miguel Ballesteros. 2018. [Pieces of eight: 8-bit neural machine translation](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 114–120, 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.
- Sujith Ravi and Zornitsa Kozareva. 2021. SoDA: On-device conversational slot extraction. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 56–65.
- Peter J Rousseeuw and Christophe Croux. 1993. Alternatives to the median absolute deviation. *Journal of the American Statistical association*, 88(424):1273–1283.
- P.J. Rousseeuw, F.R. Hampel, E.M. Ronchetti, and W.A. Stahel. 2011. *Robust Statistics: The Approach Based on Influence Functions*. Wiley Series in Probability and Statistics. Wiley.
- 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*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. [Recursive deep models for semantic compositionality over a sentiment treebank](#). In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.
- John W Tukey et al. 1977. *Exploratory data analysis*, volume 2. Reading, Mass.

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.

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](#). *CoRR*, abs/1804.07461.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. [Neural network acceptability judgments](#). *Transactions of the Association for Computational Linguistics*, 7:625–641.

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.

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*.

Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. [Q8bert: Quantized 8bit bert](#). *arXiv preprint arXiv:1910.06188*.

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. 2017. [Incremental network quantization: Towards lossless cnns with low-precision weights](#). *CoRR*, abs/1702.03044.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tiejian Liu. 2019. [Incorporating bert into neural machine translation](#). In *International Conference on Learning Representations*.

A Evaluation on GLUE Task

For GLUE experiments we use the publicly available open-source library PyTorch-Transformers (Wolf et al., 2019). We report the standard metric on each task, specifically: Accuracy is used for MNLI, MNLI-MM (mismatch) (Williams et al., 2018), SST-2 (Socher et al., 2013), QNLI (Rajpurkar et al., 2016), and RTE (Dagan et al., 2005). Mathews correlation coefficient is used for CoLA (Warstadt et al., 2019). F1 is used for MRPC (Dolan and Brockett, 2005) and QQP (Iyer et al., 2017). Finally, Pearson correlation coefficient is

used for STS-B (Cer et al., 2017), For BERT models, We use the default hyper-parameters provided by the HuggingFace’s library, specifically the learning rate is $2. \times 10^{-5}$, the batch-size is 32 and the fine-tuning epochs is 3, except for MRPC where the the fine-tuning epochs is 5. For RoBERTa models, we tuned the learning rate in $[5e-7, 2e-6]$ for best devset results on FP32 evaluations, in addition we increase the epochs to 6 for the two large datasets, MNLI and QQP, and to 12 for the rest of the tasks. Similarly to (Kim et al., 2021) we exclude WNLI (Levesque et al., 2012) since it showed unstable results even on FP32 due to its small dataset.

Fact Checking Machine Generated Text with Dependency Trees

Alex Estes*[†]
Balto, Seattle, WA
g.alex.estes@gmail.com

Nikhita Vedula*, Marcus Collins*, Matthew Cecil,
and Oleg Rokhlenko
Amazon, Seattle, WA
{veduln, collmr, mattceci, olegro}@amazon.com

Abstract

Factual and logical errors made by Natural Language Generation (NLG) systems limit their applicability in many settings. We study this problem in a conversational search and recommendation setting, and observe that we can often make two simplifying assumptions in this domain: (i) there exists a body of structured knowledge we can use for verifying factuality of generated text; and (ii) the text to be factually assessed typically has a well-defined structure and style. Grounded in these assumptions, we propose a fast, unsupervised and explainable technique, DepChecker, that assesses factuality of input text based on rules derived from structured knowledge patterns and dependency relations with respect to the input text. We show that DepChecker outperforms state-of-the-art, general purpose fact-checking techniques in this special, but important case.

1 Introduction and Background

Prior work has noted the benefits of natural language text generated by NLG models over fixed templates, for various language processing and language understanding tasks, i.e., fewer grammatical or structural disfluencies, increased conversational nature, textual diversity and user satisfaction (Kale and Rastogi, 2020; Challa et al., 2019; Vedula et al., 2023). Recent NLG approaches based on language models allow conversational agent designers to create natural-sounding responses in a very wide range of situations without exhaustively testing template-based response text (Wang et al., 2022; Asai et al., 2021; Kim et al., 2021). However, neural NLG systems may emit hallucinated, factually incorrect or even nonsensical content that is not entailed by their inputs (Wang et al., 2020; Tian et al., 2019; Liu et al., 2021b; Dhingra et al., 2019). For instance, an NLG model may synthesize multiple

independent hypotheses involving comparisons or aggregations, which must be verified. There has been growing attention towards checking the factual accuracy of statements produced by neural NLG systems in conversational settings. These statements fall into two categories depending on the source input data used for generation (Guo et al., 2022). First, the input whose factuality is to be determined can be *unstructured*, as in abstractive summarization (Teredesai et al., 2019; Goyal and Durrett, 2020) or the FEVER task (Thorne et al., 2018; Zhou et al., 2019; Zhang et al., 2020; Bekoulis et al., 2021; Vedula and Parthasarathy, 2021). Second, the input to be factually assessed may involve *structured* components, like knowledge graphs (Auer et al., 2007; Shiralkar et al., 2017) or tabular data (Chen et al., 2019; Zhong et al., 2020; Liu et al., 2021a).

In conversational, high-consequence domains like health, finance, or e-commerce, we must be sure that any factual error introduced by an NLG system is detected (Chen et al., 2021; Di Sotto and Viviani, 2022; Khan et al., 2022). While superficially similar to other fact verification efforts, this setting presents both unique challenges, and unique simplifications. In this work, we investigate the factuality of fluent e-commerce statements generated via neural NLG techniques using structured data, namely, a commercial product catalog (Ni et al., 2019) containing tables of products and their attributes. We show that the state-of-the-art systems typically developed for general-purpose structured data fail to perform well in our conversational e-commerce setting. Though we focus on the e-commerce domain, our proposed system can be used with any structured data source to check statements generated from non-e-commerce domains such as finance (e.g., the recent FinQA challenge (Chen et al., 2021) or health (consider the consequences of providing inaccurate health recommendations to users of personal health trackers

*These three authors contributed equally

[†]Work done while at Amazon

in a system like (Harris and Zaki, 2022)).

We focus on three main challenges while building our proposed fact checking model, *DepChecker*. First, since presenting factually untrue statements to users in high-consequence domains may lead to a poor decision, we must achieve very high false statement recall, and very high true statement precision. Second, deploying fact verification in a conversational setting demands speed – during runtime evaluation of responses to customers of voice assistants, and in case we want to use fact verification output as a signal to improve an accompanying conversational response generation model. As a result, we seek to avoid using large language models for fact checking in production, if possible. Third, data-to-text NLG systems may rephrase entities from the input values, e.g., if a data entity is itself disfluent. In benchmark corpora like TabFact (Chen et al., 2019), data and reference text mentions of the same entity are generally identical, which can limit the flexibility of fact verification models trained on those corpora.

DepChecker also benefits from some simplifications, especially in the e-commerce domain. We observe that in the conversational product search and recommendation setting, NLG systems are unlikely to generate product statements that have a complicated linguistic style or form (Jannach et al., 2021). Thus, unlike TabFact which requires complex hops and joins to reason over structured table data, we can concentrate on a simpler, reduced set of operations. We are inspired by (Reddy et al., 2016), who approached question answering by developing rules over dependency parse trees. Consider the example in Figure 1: key entities have very simple structures. The product mention has an *nsubj* (nominal subject) dependency¹; attributes are either *attr* (attribute) or *conj* (conjunct) dependents of *attr*. These basic observations hold over a wide set of examples in this domain, prompting us to exploit dependency trees to better extract claims, perform entity linking to match the input statement with its associated structured data, and finally determine the factuality of the statement.

A current limitation of DepChecker could be that manually constructing rules over parse trees requires both time and domain expertise. As a result, we also seek to automatically induce the manually generated rules using genetic optimiza-

tion algorithms (Mitchell, 1998), in a system we call DepCheckerGA. Our contributions can thus be summarized as follows:

1. We develop a high-speed fact-verification system that has a very high false statement recall and very high true statement precision, for future deployment in production.
2. We show that highly performing fact verification models on structured, open domain data benchmarks fail to perform well on NLG statements created from structured, domain-specific inputs, despite fine-tuning.
3. We show that our unsupervised, accurate system DepChecker is explainable and up to 10× faster than neural baselines.
4. We discuss how DepChecker’s rules can be automatically learned and generalized to other systems or domains via genetic optimization.

2 Methodology

2.1 Parsing and Fact Extraction Rules

DepChecker consists of three high-level components: (1) a set of rules applied to a dependency parse to identify the heads of entity mentions; (2) rules over token tags and text used to identify the complete mention; and (3) a system to link the identified entity mentions to corresponding structured evidence, and thereby verify whether the text is factual. Here, we focus on the product names, attribute names, and attribute values that are part of our structured product catalog. We use spaCy’s dependency parser² to extract product-attribute relations from the text. We describe several examples of rule development in Section 2.2. These rules however only identify the head-tokens of product and attribute mentions. A second set of patterns over exact strings, regular expressions, lemmas, parts of speech, and other linguistic attributes is used to complete the entity spans, described in Section 2.3. We end with product-attribute-value relations like {product_3, size, 550w} (see Figure 1), which we call a “hypothesis”. We note that an NLG based text generation model may produce non-standard statements which the parser may not parse “correctly”. We emphasize that DepChecker is not actually affected by the correctness of the dependency parse, so long as it is *consistent* across all statements in the domain. We are using the parse

¹ We use ClearNLP dependency labels: github.com/clir/clearnlp-guidelines

² spacy.io/usage/linguistic-features

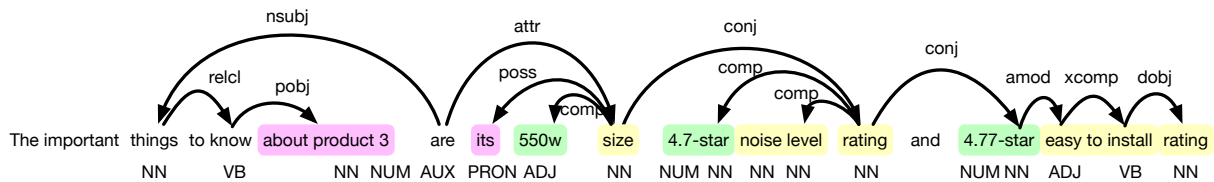


Figure 1: Identifying entity attributes from the dependency parse tree of an input claim whose factuality is to be assessed. Potential product mentions are shown in pink, attribute names in yellow, and attribute values in green.

only as a set of features to identify product and attribute mentions and their relations to each other.

Statements in this shopping domain can also compare attributes of different products. For example, for the claim “With 5 stars, Product 1 has the highest battery life rating of all three products” to be true, Product 1’s battery life rating must be 5 stars, and the other products’ battery life ratings must be less than 5 stars. We add to the hypotheses one or more *comparands* (products to which we compare), and a *comparator* (e.g., “more” or “less”). Here, there are two implied pairwise comparisons, one each with Products 2 and 3.

The third component evaluates the extracted hypotheses against the catalog data. For each hypothesis extracted from a claim, we iterate through the catalog attributes to find a match. We first check for an exact match. If there is none, we search for catalog and hypothesis attributes whose cosine similarity of their GloVe embeddings (Pennington et al., 2014) is more than 0.9. If any hypothesis fails to match, then the claim is False. If all hypotheses evaluate to True, the claim is labeled True.

2.2 Dependency Rule Development

We illustrate the process of finding good rules to identify product-attribute relationships with two examples. We annotate 250 examples and identify the dependency paths from key verbs (e.g., “has”, “is”) and possessives (poss) to product attributes. For instance, the dobj (direct object) of “has” is always a product attribute phrase. An obvious example is “Product 1 has four processors”. Now consider the example in Figure 1. We identify the verb *are*, and descend the tree from there to identify products and attributes. The first attribute (“550w size”) has a *attr* dependency arc from “are”. Remaining attributes are all (*conj*) children of another attribute. DepChecker uses six of these *attribute existence rules* to extract attributes.

Figure 2 shows a case with a possessive pronoun, *its*, whose head, *quality*, is universally an attribute,

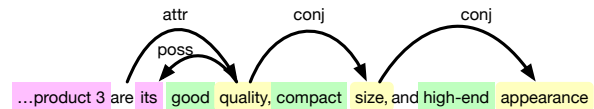


Figure 2: An example parse with a possessive, showing that attributes are the head, and conjunct dependencies of the head, of *its*. Color scheme as in Figure 1.

so we add a rule $\text{head}(\text{poss}) \rightarrow \text{attribute}$. Again, *conj* arcs link remaining attributes to the first. We follow a similar process to find the product mention, first climbing the tree to the main verb of the sentence (here “like”, not shown in the figure), and descending again to find the product mention. DepChecker has six rules for possessives. Two additional rules handle more generic verb heads, to identify product attributes phrased like “the lights are bright”. DepChecker has six more rules to handle comparisons, closely following attribute existence rules. Thus, DepChecker relies on just twenty rules to identify products from dependency trees.

2.3 Attribute Token Rules

Our dependency rules find the head token of a product or attribute, and we must identify the complete attribute and value, if present, for which we developed regular expressions on the part-of-speech tags. In all, there are 30 rules for different products and attributes. For example, to extract the complete attribute “easy to set up”, we use the pattern $/(\text{NN}|\text{ADV})^* \text{ADJ} (\text{ADP}|\text{PART}) (\text{VB}|\text{NN}) \text{NN? ADP?}/$. Finally, taking advantage of the relatively simple statement structure for the e-commerce domain, we have two co-reference rules linking pronouns to product mentions. We observe that these rules are more accurate and faster than co-reference resolution with general purpose models (Stylianou and Vlahavas, 2021).

2.4 Genetic Rule Optimization

We now describe how we can automatically learn the rules detailed in the above sections. To extract

product:attribute and *attribute name:value* pairs from statements, we first observe that we can identify a path from any token in the sentence up to a common head and down to any other token. Each pair of paths from two tokens to a shared head can be expressed as a list of dependencies. For instance, in Figure 1, “product” has a path [pobj, relcl, nsubj] to the shared head token “are”, while the attribute phrase “550w size” has a path [attr] to “are”. Together, these paths form a candidate rule for *product:attribute* extraction.

We extract all such candidate rules from a small set of ground truth-labeled example sentences. In principle, we could apply all of the extracted rules, but not all are sufficiently precise. We therefore apply genetic optimization (Fortin et al., 2012; Mitchell, 1998), optimizing our rule selection to increase product and attribute phrase head recall, while maximizing token level precision to avoid selecting spurious phrases.

We start from a set of candidate rules R_{cand} , which are regular expressions over the dependency labels linking products and attributes to common head tokens. We further define a fitness function E which we use to optimize individuals I_i , which are collections of rules, and the population P as a whole. In our case, we optimize for a combination of precision and recall in identifying labeled products and attributes, and add a penalty for adding more rules. To initialize the population, we create n_{pop} individuals, each of which is initialized by randomly selecting between 1 and 8 rules from R_{cand} . Next, we begin to evolve the population. At each step, we evaluate each individual in the population using E . While our best individual is below some target threshold fitness E_{tgt} , we select individuals, mutate some of them, and recombine rules between individuals.

Selection consists of choosing the most “fit” individuals from the population, based on the fitness evaluated using E . We use tournament selection (Mitchell, 1998), implemented using `deap` (Fortin et al., 2012)³. In tournament selection, we choose a tournament size k (10 in our case) and select that number participants randomly from the current population generation. The most fit individual from that subset of k individuals is added to the next generation. We use sampling with replacement, so the most fit individuals are likely to be added more than once to the next generation, strengthening the next

generation while also allowing for some variation to carry forward.

Next, we iterate over all individuals, first removing from each individual any redundant rules which are wholly entailed by some other rule. Then with probability p_{mut} , we mutate the individual, and recombine or “crossover” rules with another individual with probability p_{cr} . We mutate individuals to diversify the new rule generation. Then, we merge rules. Merging consists of aligning pairs of rules and combining the aligned forms. We use the Needleman-Wunsch (Needleman and Wunsch, 1970) algorithm to align the patterns. We choose the most closely aligned pairs of rules to merge. Where the tokens match, we simply add them to the pattern; where there are gaps, we add a ‘*’ (repeater or Kleene star) operator, and where there are differences, we use an ‘|’ (or) operator. For example, given the patterns ABCD and ACE, the alignment and merge would result in AB*C(D|E). As we will explain below, this pairwise alignment is somewhat limited, and can lead to complex patterns that are hard to further combine or generalize.

If we recombine rules with another individual, we first select an individual at random. We then choose a rule at random from each individual, and swap them between the two individuals.

3 Evaluation

To test our proposed approaches, DepChecker and DepCheckerGA, we first developed a BART-based (Lewis et al., 2020) data-to-text NLG model to generate conversational statements in the e-commerce domain, taking as input tabular product-attribute data from a structured product catalog (Ni et al., 2019). To train this NLG model, we asked human annotators to write text describing and comparing the product attribute values of 2-3 catalog products (Figures 1 and 2 contain examples of the human generated sentences). We used this set of NLG model generated sentences as our test set for factuality assessment, to test DepChecker in this work. As a baseline for DepChecker, we also trained a supervised factual error detection binary classifier (henceforth called RoBERTaChecker) using the RoBERTa (Liu et al., 2019) language model, with a mix of the above mentioned human-generated product domain statements and artificially generated negative examples (generated by corrupting the entity names and values in the human-generated factually correct statements). This binary classifi-

³ `deap.readthedocs.io`

Method	Acc	TP	FR	AIT
Table-BERT (Chen et al., 2019)	0.56	0.50	0.31	452
GNN-TabFact (Ran et al., 2019)	0.43	0.53	0.20	644
TAPEX-base (Liu et al., 2021a)	0.52	0.54	0.79	228
TAPAS-base (Herzig et al., 2020)	0.51	0.66	0.97	724
RoBERTaChecker	0.67	0.87	0.94	681
DepChecker (proposed)	0.65	0.85	0.94	72
DepCheckerGA (proposed)	0.63	0.84	0.94	256

Table 1: Fact verification performance of different models. **Acc**: Accuracy; **TP**: True precision; **FR**: False recall; **AIT**: Average Inference Time (in milliseconds).

cation dataset was also used to fine-tune the neural baselines shown in Table 1. The first four baselines are approaches that have been specifically developed for the TabFact dataset (Chen et al., 2019), and were also fine-tuned on TabFact. We also labeled some of this data with product name, attribute name, and attribute value tags, to select dependency rules as described in Section 2.4.

The factual error detection test set to evaluate DepChecker consists of 195 sentences generated by our BART based NLG model described above, and includes 98 factually consistent or true statements and 97 factually inconsistent or false statements. This test set was manually labeled after finalizing the DepChecker rules, to avoid any data leakage. The values of all hyperparameters used by DepChecker and DepCheckerGA were tuned on a manually curated validation set. We report the average execution time over all examples at inference, on a single 2.3GHz Intel Xeon E5-2686 v4 CPU (an Amazon AWS EC2 p3.8xlarge instance). Note that we explicitly include the dependency parsing time in the reported execution time for DepChecker. An example of a structured product table from our dataset is `{size_product3: 550w, noise_level_rating_product3: 4.7, easy to install rating_product3: 4.77}`, from which our BART based NLG model generates the sentence shown in Figure 1.

3.1 Results

As shown in Table 1, DepChecker outperforms all baselines on accuracy or speed, and outperforms most models on both. Models developed for large-scale fact checking baselines have surprisingly low accuracy: the best system, TAPAS (Herzig et al., 2020) reaches only 51%: it labels nearly all examples as false. A more important task here is to detect false statements, since making a false statement to a customer in an e-commerce setting could seriously damage a business’s reputation. TAPAS identifies 97.0% of factually inconsistent

sentences (at the cost of very low precision on factually true statements), followed by our baseline neural model, RoBERTaChecker, with false recall 94% but much better overall accuracy 67%. DepChecker is equally good, reaching 94% false recall and 65% accuracy. In addition, neural models, especially those using transformer architectures, are slow. Using GPUs makes them faster, but much more costly. For both cost and energy impact in a production deployment setting, as well as for a lower latency in a conversational response generation setting, it is desirable to use more efficient models. Table 1 shows that DepChecker is three to ten times faster than all other baseline models, and 9.4 times faster than the only neural model with comparable accuracy, RoBERTaChecker.

We next test how well we could automatically reproduce DepChecker’s rules using our genetic optimization scheme in Section 2.4 (*DepCheckerGA*). We find that the rules generated by genetic optimization are not as compact or efficient as manually created rules. These rules extract 95% of head tokens of the 1000 product name, attribute name, and attribute value spans in our test set, and 96% of the extracted tokens belonging to those spans. These values are statistically indistinguishable from DepChecker, albeit with almost twice as many rules. As a result of the additional rules, average execution time increases to about 250 milliseconds per sentence⁴, making DepCheckerGA still superior to several of our baseline systems.

3.2 Discussion

Error Analysis and Explainability: Of all models, DepChecker and RoBERTaChecker have significantly higher accuracy and high false recall. They mis-classify very few false statements, and also make different errors. Combined, these two models achieve 99% false recall, better than any other system. RoBERTaChecker makes errors on encountering attributes altered from the input table (“80 plus gold” → “80 plus gram”), misses negation (attribute “heat sensitive: false” → “customers like its heat sensitivity”, or when an attribute value is associated with the wrong product. DepChecker makes errors only of the first kind, *e.g.*, if it gets as input “soft material” for the attribute “soft”.

Other neural models appear to make similar er-

⁴ Note that to implement our genetic optimization, we convert dependency parses to strings and execute regular expression rules; it may be possible to further optimize this by converting the rules to SpaCy’s tree regex, used in DepChecker.

rors to RoBERTaChecker. For instance, TAPEX classifies "... product 3's 1.0-inch unit count ...", as true even though it is obviously wrong; DepChecker classifies it correctly. TAPEX and TAPAS both fail in cases where NLG transforms attributes like "battery" to "battery life", or when the NLG fabricates an attribute completely, with no corresponding input value. As with RoBERTaChecker, TAPEX and TAPAS make orthogonal errors to DepChecker, and near 100% false recall can be achieved. We aren't sure why this should be, since the neural models are opaque. However, unlike other state-of-the-art systems, our proposed approach DepChecker is both fast and explainable. Its rules make it clearer to diagnose and understand the exact reason DepChecker fails on an input, and use this signal to correct the input claim or improve the fact verification process. DepChecker's speed also enables it to be easily used in real-time systems, and as a potential signal to improve the training of NLG models generating factually inconsistent statements.

Comparing DepChecker and DepCheckerGA rules: We assess whether DepCheckerGA can learn similar rules as those created by humans or expert linguists (see Table 2). First, we observe DepCheckerGA's rules tend to be more narrow, illustrated by the second example in Table 2, which has a number of additional constraints compared to the DepChecker rule, which has broader coverage as well. This is likely due to the fact that in order to facilitate the genetic optimization strategy, DepCheckerGA must fully specify the dependency paths to each product and attribute in its rules. By contrast, a linguist can find patterns that involve only the key parts of the dependency links. It may also be partly due to the limitations of our alignment and merging strategy in Section 2.4. This leads to more, and unnecessarily specific rules.

In many cases, however, the rules found by DepCheckerGA match those formed by expert linguists. The first and third examples in Table 2 demonstrate this. However, we note that DepCheckerGA has redundant rules that identify different parts of the same sentences. In the third example, the rules shown find a link between "product", "strap", and the shared head "style". Another DepCheckerGA rule, NOUN conj* ((pobj prep) | amod)⁵, unnecessarily identifies "adjustable", showing that DepCheckerGA

could also do a better job pruning its rules.

Finally, while matches are more or less uniformly distributed across our hand-crafted rules, only about 1 in 6 of DepCheckerGA's rules match more than 10 statements. In future, we will further explore DepChecker's rule combining strategies for it to match DepChecker's performance. Further, our rule merging relies on two rules being mostly identical. We do not merge multiple rules at once to generalize better. Multiple sequence alignment techniques could be used to improve DepCheckerGA (Chowdhury and Garai, 2017).

Dependency Parsing Errors: As noted in Section 2.1, since we learn dependency patterns from the parsed NLG text, the parse need only be *consistent* across the domain, and not linguistically correct. Parsing errors are taken into account while learning the rules of DepChecker. It thus implicitly handles any language errors made by the chosen dependency parser, which might be more common if the NLG input is not completely fluent. For instance, in the statement *customers feel highly positive about product 1's great features*, the parser incorrectly labels "features" as the object of adposition *about*, when it should be *product*. Both DepChecker and DepCheckerGA learn rules to handle this technical error.

Generalizability: We showed that we can automatically learn a comparable set of rules over dependency parse trees using genetic optimization (Section 2.4). This allows DepChecker to generalize to other domains, requiring only a small set of data instances labeled with the relevant entity names and values. It also allows a domain-specific fact verification system to be built rapidly with almost no manual effort. However, as mentioned earlier, the automatically learned rules may be less efficient, so some expert linguist effort may still be required. Our analysis indicates that our domain of conversational shopping likely exhibits a comparatively small amount of variation in style and linguistic structure of the text whose factuality is being assessed in this paper. Extending DepChecker to work well for domains with higher textual variation is something we leave for future work.

To investigate DepChecker's generalizability to large-scale, general purpose corpora like TabFact, we extend our defined rules to a sample of TabFact entities, which have been derived from DBpedia (Auer et al., 2007). We replace the 'product' and 'attribute' entity mentions in our rules by more

⁵ see Table 2 caption for notation

	attribute rule; product rule <i>or</i> example sentence
DepChecker DepCheckerGA <i>example sentence</i>	<i>have</i> dobj ; <i>have</i> nsubj <i>have</i> dobj ; <i>have</i> nsubj conj* Product 2 has the highest rating .
DepChecker DepCheckerGA <i>example sentence</i>	<i>have</i> dobj .* conj ; <i>have</i> nsubj <i>have</i> dobj ; VB advcl (acompl dobj*) conj (acompl amod*)* According to reviews, previous buyers feel positive about product 1 because it has easy setup and installation , great picture quality , and lights are bright .
DepChecker DepCheckerGA <i>example sentence</i>	<i>attribute</i> <i>product</i> case; <i>attribute</i> .* conj <u>NOUN</u> conj *; <u>NOUN</u> (nmod poss) previous buyers feel highly positive about product 1’s laptop <u>style</u> , adjustable strap , and 17.3-inch grey color .

Table 2: Comparing similar rules from DepChecker (manually created) and DepCheckerGA (automatically learned via genetic optimization). Rules are expressed as regular expressions over dependency paths. Italics indicates a specific lemma must be at that position. Capital letters indicate a part-of-speech class. Heads are to the left. We use universal dependencies and standard regular expression notation: * means zero or more instances of the token, '.' means any token. Bolded words indicate matched attributes; underlined words indicate matched heads. Note that *attribute* stands for any word, and *product* is a specialized lemma.

generic ‘table-row’ and ‘table-column’ tokens of the TabFact tables, and verify if a claim is true, given an input table. DepChecker cannot yet handle some of TabFact’s aggregations, so we check the statement veracity for each table row independently. If a claim is false for any one row of the input table, we label it False, otherwise it is labeled True. This simple technique achieves 60% accuracy on the TabFact test set (current state-of-the-art model accuracy is 85%), suggesting the potential of using dependency parse structure in more general or complex fact checking scenarios.

4 Limitations

The major limitation of our work is that DepChecker in its basic form requires manual effort to construct rules for fact checking. While we showed that this rule construction can be automated, it comes with a trade-off in the compactness of the automatically generated rules and the overall latency. However, as shown in Sections 3.1 and 3.2, large, neural language models developed for fact checking with structured general purpose data (e.g., TabFact derived from Wikipedia tables) are unable to perform fact checking well on statements based on our e-commerce structured catalog data. The time required to assemble such a large volume of labeled data in specific domains (e.g., health or e-commerce) to train large neural models would certainly cost more than one or two weeks of a linguist’s time, and may still not be fast or accurate enough to be deployed for production purposes.

Hence, it isn’t clear how much effort would be required to build a neural system comparable in speed and accuracy to DepChecker, or that the benchmark systems scale or generalize better than DepChecker for a real-world deployment scenario. Like existing data-to-text fact checking work, DepChecker also assumes that the underlying structured data used to verify the statement factuality is accurate.

Finally, we acknowledge that rule-based approaches are not new, but as we show in this work, they can still outperform or at least complement state-of-the-art language models. Their speed, simplicity, accuracy, and explainability make them valuable tools in production use cases. Since seemingly state-of-the-art results based on large language models may not generalize as well as hoped across different domains and datasets, we would like to point out that our rule-based approach, DepChecker, remains competitive in many scenarios. Such a system can therefore still be worth investing in, especially in an age of larger and slower models.

5 Conclusion

We propose DepChecker: a fast, unsupervised, and explainable model to detect factual inconsistencies in a conversational shopping and recommendation setting. We show that DepChecker can outperform strong baselines, and suggest that instead of pursuing costlier, less explainable, slower models, more research should go into how to further automate development of simpler, less expensive and more interpretable models for production use cases.

References

- Akari Asai, Matt Gardner, and Hannaneh Hajishirzi. 2021. Evidentiality-guided generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2112.08688*.
- Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.
- Giannis Bekoulis, Christina Papagiannopoulou, and Nikos Deligiannis. 2021. A review on fact extraction and verification. *ACM Comput. Surv.*, 55(1).
- Ashwini Challa, Kartikeya Upasani, Anusha Balakrishnan, and Rajen Subba. 2019. Generate, filter, and rank: Grammaticality classification for production-ready nlg systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 214–225.
- 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. In *International Conference on Learning Representations*.
- Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan Routledge, and William Yang Wang. 2021. **FinQA: A dataset of numerical reasoning over financial data**. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Biswanath Chowdhury and Gautam Garai. 2017. A review on multiple sequence alignment from the perspective of genetic algorithm. *Genomics*, 109(5-6):419–431.
- Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.
- Stefano Di Sotto and Marco Viviani. 2022. Health misinformation detection in the social web: An overview and a data science approach. *International Journal of Environmental Research and Public Health*, 19(4):2173.
- Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175.
- Tanya Goyal and Greg Durrett. 2020. **Evaluating Factuality in Generation with Dependency-level Entailment**. *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3592–3603.
- Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.
- Jonathan Harris and Mohammed J. Zaki. 2022. **Towards neural numeric-to-text generation from temporal personal health data**. *arxiv*.
- Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, 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.
- Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)*, 54(5):1–36.
- Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520.
- Suleman Khan, Saqib Hakak, N Deepa, Kapal Dev, and Silvia Trelova. 2022. Detecting covid-19 related fake news using feature extraction. *Frontiers in Public Health*, page 1967.
- Jinhyeon Kim, Donghoon Ham, Jeong-Gwan Lee, and Kee-Eung Kim. 2021. End-to-end document-grounded conversation with encoder-decoder pre-trained language model. In *Proceedings of the DSTC9 Workshop, Online*, pages 8–9.
- 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.
- Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021a. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.
- Tianyu Liu, Xin Zheng, Baobao Chang, and Zhifang Sui. 2021b. Towards faithfulness in open domain table-to-text generation from an entity-centric view. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13415–13423.

- 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*.
- Melanie Mitchell. 1998. *An Introduction to Genetic Algorithms*. MIT Press.
- Saul B. Needleman and Christian D. Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.
- Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. 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 188–197.
- 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.
- Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. Numnet: Machine reading comprehension with numerical reasoning. 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 2474–2484.
- Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.
- Prashant Shiralkar, Alessandro Flammini, Filippo Menczer, and Giovanni Luca Ciampaglia. 2017. Finding streams in knowledge graphs to support fact checking. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 859–864. IEEE.
- Nikolaos Stylianou and Ioannis Vlahavas. 2021. A neural entity coreference resolution review. *Expert Systems with Applications*, 168:114466.
- Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, George Karypis, Ben Goodrich, Vinay Rao, Peter J Liu, and Mohammad Saleh. 2019. Assessing The Factual Accuracy of Generated Text. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 166–175.
- 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.
- Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation. *arXiv*.
- Nikhita Vedula, Marcus Collins, Eugene Agichtein, and Oleg Rokhlenko. 2023. Generating explainable product comparisons for online shopping. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*.
- Nikhita Vedula and Srinivasan Parthasarathy. 2021. Face-keg: Fact checking explained using knowledge graphs. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 526–534.
- Weizhi Wang, Zhirui Zhang, Junliang Guo, Yinpei Dai, Boxing Chen, and Weihua Luo. 2022. Task-oriented dialogue system as natural language generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2698–2703.
- Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. 2020. Towards Faithful Neural Table-to-Text Generation with Content-Matching Constraints. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086.
- Wenxuan Zhang, Yang Deng, Jing Ma, and Wai Lam. 2020. AnswerFact: Fact checking in product question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2407–2417, Online. Association for Computational Linguistics.
- 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.
- Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2019. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 892–901, Florence, Italy. Association for Computational Linguistics.

Prototype-Representations for Training Data Filtering in Weakly-Supervised Information Extraction

Nasser Zalmout and Xian Li

Amazon.com

{nzalmout,xianlee}@amazon.com

Abstract

The availability of high quality training data is still a bottleneck for the practical utilization of information extraction models, despite the breakthroughs in zero and few-shot learning techniques. This is further exacerbated for industry applications, where new tasks, domains, and specific use cases keep arising, which makes it impractical to depend on manually annotated data. Therefore, weak and distant supervision emerged as popular approaches to bootstrap training, utilizing labeling functions to guide the annotation process. Weakly-supervised annotation of training data is fast and efficient, however, it results in many irrelevant and out-of-context matches. This is a challenging problem that can degrade the performance in downstream models, or require a manual data cleaning step that can incur significant overhead. In this paper we present a prototype-based filtering approach, that can be utilized to denoise weakly supervised training data. The system is very simple, unsupervised, scalable, and requires little manual intervention, yet results in significant precision gains. We apply the technique in the task of attribute value extraction in e-commerce websites, and achieve up to 9% gain in precision for the downstream models, with a minimal drop in recall.

1 Introduction

Weak supervision and data programming have recently emerged as powerful techniques to support information extraction models. Weak supervision is useful for dynamic environments, where new tasks or deployment domains keep emerging, and using manually annotated data is impractical. Weakly supervised data programming aims to reformulate the training data annotation process into a programming paradigm. Instead of manually annotating each training sample, the annotation process is handled through labeling functions, which are then used to automatically annotate the training corpora. These labeling functions are usually based

on manually predefined patterns or regular expressions, that are matched against the target unannotated data. The weak supervision setup allows for an efficient and scalable training data collection process, but at the expense of accuracy. Labeling functions can over-match, resulting in many irrelevant and out-of-context matches. This hinders the performance, and could require manual cleaning steps to lift quality, adding to the overall overhead.

In this paper we present a data filtering system based on prototype-learning. *Prototype* here refers to the correct contexts where a given target value is usually mentioned, based on a sample dataset. This can be drawn from a manually annotated corpus (making it more supervised), or it can be based on the centroid of the raw dataset (more noisy, but totally unsupervised). At runtime, we get the embedding of the prototype, along with context embedding of the annotated label for each training sample, and use outlier detection constructs to remove outliers. We calculate the distance between the prototype and the in-context label representation, and if the distance is above a certain threshold we filter it out. These out-of-context annotations are out-of-distribution, so approaches that utilize prototype-learning as part of the end-to-end network, or other noise-robust strategies, would not handle them properly. Removing them before modeling is the best approach to reduce noise. The intuition behind this filtering logic is that value context embeddings that are distant from the prototype tend to be more noisy, likely reflecting out-of-context matches. We use several prototype and data embedding techniques.

We utilize the filtering technique for attribute value extraction (AVE), an information extraction task that has recently been gaining much momentum in e-commerce applications. The goal of the AVE task is to obtain structured product features from the unstructured natural language description of the product's page in e-commerce websites. This

Desert Essence Island Mango Hand & Body Lotion - 8 Fl Ounce
 - Enriching - Aloe Vera - Jojoba & Coconut Oil - Shea Butter -
 Delightful Scent - Moisturizes & Refreshes Skin



About this item

- **MOISTURIZES SKIN** - Organic shea butter and jojoba oil are used in the Island Mango hand and body lotion to moisturize the skin.
- **SILKY SMOOTH SKIN** - The blend of coconut oil and jojoba oil in this hand and body lotion nourishes your skin and effectively makes it smooth and silky.
- **CAPTIVATING SCENT** - Mango seed butter and passion flower extract are used in the body lotion for a delightful scent that captivates the senses.
- **ORGANIC INGREDIENTS** - The Island mango hand and body lotion is made using organic ingredients like coconut oil, shea butter, carnauba wax, jojoba seed oil, and sunflower seed oil.

Attributes: Brand Scent Item Form Size Benefit

Figure 1: Sample product profile and relevant attributes.

has several downstream applications in areas including product search, comparison, question answering, among others. Due to the dynamic and large scale nature of this domain, contributions for the AVE task often use weak and distant learning techniques to train the extraction models. Our results show up to 9% absolute improvement in precision, with minimal drop in recall (about 1%), for the extraction model trained on the filtered data.

2 Background

2.1 Attribute Value Extraction

The AVE task aims to extract the corresponding values for a given attribute, out of a number of attributes of interest, from the textual sequence of a product profile (Zheng et al., 2018). Given a text sequence $X = [x_1, \dots, x_n]$ in a product profile, where n is the number of words, and an attribute $r \in R$, where R is a predefined set of attributes, the model is expected to extract all text spans from X that could be valid values for attribute r characterizing this product’s features. When there are no corresponding values mentioned in X , the model should return an empty set. For example, for the product in Figure 1, given its title as X , the model is expected to return (“8 Fl Ounce”) if $r = \text{“Size”}$, and an empty set if $r = \text{“Hair Type”}$. The various products are categorized into diverse product types (PTs), like *Shampoo*, *Tea*, *TVs*, etc. The products within a given PT are homogeneous, sharing similar overall product features, including the set of relevant attributes. And different PTs can have a different set of relevant attributes.

The content in e-commerce websites is very dynamic, where new products are frequently added.

The PT categorizations themselves also change overtime with new products, along with the set of relevant attributes. This makes manual training data annotation with a predefined set of PTs and attributes infeasible for attribute value extraction. Zero and few-shot learning for new attributes have also shown limited success (Yang et al., 2022). Therefore, most of the AVE contributions rely on distant or weak supervision, which is very susceptible to noisy and out-of-context annotations.

2.2 Training Data Denoising

To better understand how weak supervision with simple regular expressions or labeling functions can result in very noisy annotations, we use the product in Figure 1 as an example. The figure shows the product profile, which includes the title, description, along with the product image, for a *Skin Moisturizer* PT. Some of the relevant attribute values are highlighted with different colors. These are the values that an AVE model is expected to return, out of the target space of each separate attribute and PT. The target space for the *ItemForm* attribute and *Skin Moisturizer* PT, for example, includes “lotion”, “cream”, “oil”, among others. The *ItemForm* label for this product should be “lotion” as shown. However, a weakly supervised training data, that does not consider contextual understanding when assigning labels, could have chosen the more frequent “oil” as the label. The same behavior could happen with the *Scent* attribute, where “Coconut” could have been selected instead, or in addition to, “Island Mango”, which is the right value for this product. This paper suggests a filtering step on top of the weakly supervised training data, that eliminates such out-of-context annotations.

Our filtering setup is on the <attribute, PT> pair level. So we learn a different prototype for each <attribute, PT> pair, and apply the filtering approach for product attribute values in each pair separately.

2.3 Desiderata

There are a few constraints and specific desiderata that should guide the filtering technique to be deployed actively in a production system, without being disruptive to the advantages of distant-supervision. The filtering approach should be unsupervised, with minimal manual overhead. Ideally, the filtering approach should also provide an easy mechanism to control the balance between precision and recall, if needed. And finally, the filtering

approach should be easy to deploy without causing much disruption to existing pipelines.

3 Related Work

Prototype-based approaches for NLP have traditionally been used in the word representation literature (Huang et al., 2012; Reisinger and Mooney, 2010). Interest for prototype-based approaches increased significantly with the onset of Prototypical Networks (ProtoNet), with successful utilization in few-shot learning classification in computer vision tasks (Snell et al., 2017), and contrastive learning models (Gao et al., 2021). ProtoNet-based approaches compute one prototype per class as the class mean. These prototypes are then used in a nearest neighbour classifier to update the objective function. This is consistent with our overall setup. There have been many contributions since then utilizing Prototypical Networks for NLP tasks, mostly in few-shot learning in information extraction (Cui et al., 2021; Lai et al., 2021; Gao et al., 2019). But as far as we are aware, this paper is the first to use prototype representations for training data filtering in weak supervision models. There are other contributions utilizing data centroids for outlier detection, mostly through k-nearest neighbor formulation as well. The idea is to remove samples that are far from the cluster’s centroid, in a clustering setup (Wang et al., 2021; Pamula et al., 2011). And we in fact use a similar formulation in our filtering approach, with the context prototype as the target.

The attribute value extraction task has traditionally been modeled using distant-supervision (Ding et al., 2022; Yang et al., 2022; Lin et al., 2021; Yan et al., 2021; Wang et al., 2020; Zheng et al., 2018) which is prone to noise. Practical utilization of the AVE task in production makes further use of weak-supervision and data programming techniques for training data collection (Zalmout et al., 2021). This further amplifies the noise issue, and makes training data filtering techniques more important.

4 Approach

4.1 Context-Aware Embeddings

Each product in a given <attribute, PT> pair is represented through the context embedding of the mentioned attribute value, using pre-trained language models like BERT (Devlin et al., 2018). We use a masking vector on top of the text sequence, for each value. We then use BERT-like models to get the context embedding with the sequence and

mask as input. Within this scope, we can use two embedding paradigms:

- Value-Based Embeddings: We get the context embedding for the value mention in each product profile directly, based on the target value. For multi-worded values, we take the average of the word embeddings.
- Name-Based Embeddings: We replace the value mention with the attribute and PT names, separated with [BOA] and [EOA] special tokens (BOA: beginning of attribute, EOA: end of attribute). Like “[BOA] skin moisturizer item form [EOA]”, instead of the “*lotion*” value in the example in Figure 1. We then get the embedding as before.

We also fine-tune the pre-trained language model using the MLM objective. Fine-tuning is more critical for name-based embeddings, since the pattern of using the attribute and PT names instead of values, along with the additional special tokens, are not covered in the existing pretrained models. We follow the same value format mentioned above, and replace the attribute value with the PT and attribute names, along with the [EOA] and [BOA] special tokens. In the fine-tuning dataset, we randomly replace value mentions with the above name notation for $n\%$ of the overall corpus products.

It is worth noting that the masking setup in the name-based embeddings is used both during fine-tuning and context embedding retrieval at runtime.

4.2 Prototype Representation

The prototype embedding is the mean of the context embeddings of a representative sample of the values in a given <attribute, PT> pair. The prototype representation in prototype-learning is usually learnt from a small set of manually annotated data. However, in our case having annotated data for each PT is challenging, since production systems would be working with a large number of different PTs. And collecting annotated data for each PT is expensive. We therefore identify two different approaches to obtain the prototype representation; using a small annotated sample as typically done in classical prototype-learning, or using the centroid of the raw training data as a proxy for the prototype.

4.2.1 Gold Prototype

Manually annotated data for each PT would allow the model to capture more representative embed-

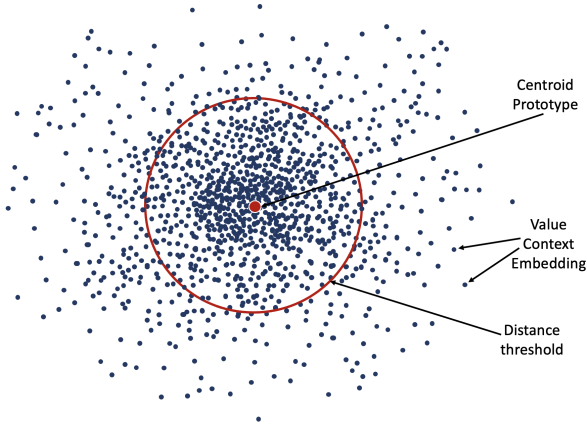


Figure 2: Filtering setup using centroid prototype.

dings. In this case, the PT embedding might not be at the centroid of the training data cluster, depending on how noisy and how representative it is of the real distribution. The main advantage of this setup is that we can have more accurate and meaningful representations, for the PTs with enough annotated data coverage. However, the size of the annotated sample for each PT would be small, given the large number of different PTs. This might lead to bias, misrepresentation, and limited coverage of all possible target values. Moreover, relying on annotated data incurs significant overhead, and creates a dependency between data annotation and the training data generation process, which hinders scalability.

4.2.2 Training Data Centroid Prototype

We also use the training data itself to calculate the prototype. The training data is noisy, so it cannot be used directly to get the prototype. However, in this case the goal would be to identify the centroid of the context embeddings, and then assign a numeric distance score for each context against the centroid as a proxy. The threshold for the distance is then used to eliminate outlier contexts relative to the centroid representation. This can work if the training data is not too excessively noisy, where the centroid is somewhat close to the PT representation if a large amount of gold data was used. The advantages of such setup is that the weakly supervised training data is cheap and does not require manual curation. We can also get sizable training data for each PT, that covers most of the relevant values. However, if the training data is too noisy, the centroid would not be capturing any meaningful representations. Figure 2 shows a sample distribution of the different contexts, centroid, and distance threshold.

4.3 Outlier Detection

After obtaining the prototype representation, whether using the gold prototype or data centroid, along with the individual context embeddings, we formulate the cleaning task as an outlier detection task. We use distance metrics to calculate distance between each training sample context embedding and the prototype. And eliminate training samples with a distance above a tunable threshold. We experiment with several distance metrics, in addition to Euclidean distance, including:

- **Mahalanobis Distance:** Mahalanobis distance is a multivariate distance metric, that considers the potential covariance between the different variables. It is commonly used in anomaly detection literature. However, excessive noise can bias the covariance matrix, hence resulting in biased distance calculations.

$$d(x, \mu) = \sqrt{(x - \mu)^T C^{-1} (x - \mu)}$$

Where x is the vector representation of the given sample. μ is the vector representation of the centroid or prototype, C^{-1} is the inverse covariance matrix estimate for the samples.

- **Cosine Distance:** The inverse of the cosine of the angle between sample and prototype vectors, through the dot product of the vectors divided by the product of their lengths.

$$d(x, \mu) = 1 - \frac{\sum_{i=1}^n x_i \mu_i}{\sqrt{\sum_{i=1}^n x_i^2} \sqrt{\sum_{i=1}^n \mu_i^2}}$$

4.4 Evaluation Criteria

Throughout the various experiments we evaluate the filtering setup based on two different criteria, precision/recall for the training data itself, and precision/recall for the downstream extraction model.

Training data recall vs model recall. We optimize mainly for precision in the training data evaluation. Recall in the training data evaluation is calculated based on the intersection of the benchmarking and training datasets, and does not necessarily correlate with the recall of the model itself. Therefore, lower recall in the training data evaluation is not problematic, as long as it does not cause a significant bias in specific values, as in eliminating certain values completely or near completely. Whereas for the actual trained model, we optimize

Attribute	# PTs	Training Set	Gold Testing Set
ContainerType*	34	265,822	2,119
ItemForm*	14	1,195,256	5,432
Pattern	33	126,153	2,665
ItemShape	80	511,977	2,825
ChocolateType	1	6,797	75
Material	21	41,245	2,198
ControlType	9	72,489	892
Sum	192	2,219,739	16,206

Table 1: Statistics of the training and evaluation datasets for the various attributes. *We use two attributes (ItemForm and ContainerType) for the ablations. And we expand to the remaining attributes afterwards.

for both precision and recall. The goal is to maximize precision, with minimal sacrifice in recall. We confirm this behavior in Table 4, where the recall of the model does not drop significantly, even though the training data evaluation reflects a bigger drop. Throughout the training data evaluation experiments we mainly focus on the precision results, but also report recall as a sanity check. But we opt not to report F1 scores, since it does not reflect a meaningful metric in this case.

5 Experiments and Results

5.1 Dataset

We collect our raw dataset from the product profiles (title, bullets, and description) from the public web pages at [Amazon.com](https://www.amazon.com). The goal is to collect training data that is entirely weakly supervised, without any manual cleaning or intervention. This is why we opted not to use available public data like MAVE (Yang et al., 2022), which has been extensively processed. We selected seven different attributes, and identified the set of relevant values. The value identification is the only manual step in this setup, gathered from Amazon pages. The training data is then collected through labeling functions based on regular expressions for each of the target values. This setup is commonly used in the AVE task, usually followed by a manual curation step to fix erroneous matching patterns.

To better understand the limits of our setup, we also worked on enhancing the quality of the training data manually, to compare against the automatic filtering system. We selected a sample of products per PT, and worked with annotators to identify patterns of erroneous value annotations in the training data and fix them. The goal is to update the labeling function with negative patterns that it should avoid matching, through look ahead

and behind phrases in the regular expression. For example, "whole" is a valid value for the ItemForm attribute, used in cases like "whole beans". A negative matching pattern would be phrases like "whole foods". A manually curated pattern in this case is to avoid matching the "whole" value if it is followed by the "foods" word. We call the resulting dataset *manually curated* throughout the experiments.

We also collected a benchmarking set of manually annotated set of products in each PT, for general evaluation. Table 1 shows the statistics of the datasets we collected. We also collected a dataset of about 3 million products, from the public pages at Amazon.com. We use this dataset for the pre-trained language model fine-tuning, using the MLM objective.

5.2 Training Data Evaluation

In this part we evaluate the resulting training datasets directly, through a manually labeled sample from the raw training data. Since the various ablations aim to filter out erroneous annotations, the recall of the raw data would be the upper bound for all subsequent variations. Recall of training data is not as important as precision, since the downstream extraction model is expected to cover the recall gap, as we highlighted earlier. So we report recall in the various results, but we focus on precision gain. We use the raw data as the main baseline. We also compare against the manually curated datasets, that were handled through manual inspection and sets of manually curated rules to fix them. The K value, at the P@R=K metric, were chosen for each case to match the recall for the manually curated datasets, to facilitate easier comparison.

Results in Table 2 show significant improvement compared to the unfiltered data, along with large improvements compared to the manually curated data as well. Value-based embeddings outperform name-based embeddings across the various settings. And the centroid approach seems to outperform the Gold Prototype approach. This is significant, since it indicates that we do not need manually annotated dataset to utilize the filtering approach. This is probably due to the more representative nature of the centroid, despite the noise, compared to a small annotated sample. We also experiment with the different outlier detection methods. Results in Table 3 show that Cosine distance outperforms the other metrics. One theory for why Mahalanobis distance did not perform well is that covariance

Training Data		ItemForm		ContainerType	
		Precision	Recall	Precision	Recall
Raw Data		86.5%	44.0%	78.1%	24.1%
Manually Curated		93.2%	29.6%	80.3%	20.2%
Centroid Prototype	Value-based max precision	96.4%	21.2%	82.7%	18.8%
	Value-based P@R=K*	95.6%	30.0%	81.6%	20.0%
	Name-based max precision	89.2%	38.0%	80.0%	21.0%
	Name-based P@R=K*	88.5%	30.0%	77.6%	20.0%
Gold Prototype	Value-based max precision	95.1%	21.5%	81.4%	18.5%
	Value-based P@R=K*	94.7%	29.3%	80.5%	20.4%
	Name-based max precision	87.1%	38.2%	80.5%	21.5%
	Name-based P@R=K*	84.5%	30.5%	75.4%	19.4%

Table 2: Training data evaluation results after the various filtering approaches. These results reflect the training data evaluation, not the extraction model evaluation. Therefore, precision gain is more important than recall, and raw data recall is not a baseline. Check Section 4.4 for more details. *The K value, for P@R=K, is 0.3 for ItemForm, and 0.2 for ContainerType, as described in Section 5.2.

Distance Metric	ContainerType		ItemForm	
	Precision	Recall	Precision	Recall
Raw Data	78.1%	24.1%	86.5%	44.0%
Cosine Distance	81.6%	20.0%	95.6%	30.0%
Mahalanobis Distance	78.6%	20.8%	88.4%	30.9%
Euclidean Distance	78.9%	20.3%	92.7%	29.8%

Table 3: Training data evaluation results for the various distance metrics. The data is filtered using value-based embeddings, centroid prototype, and the P@R=K setup. We do not include F1 results, as explained in Section 4.4.

matrices are susceptible to noise. To test if this is more prominent in centroid-based filtering, we also used the Gold Prototypes approach, and results are actually lower, in accordance to the results for Gold Prototypes in general.

We also experimented with a number of pre-trained language models, including BERT (Devlin et al., 2018) (base and large), RoBERTa (Liu et al., 2019) (base and large), and GPT2, all fine-tuned using the same dataset. Results are very close to each other, besides GPT2 which is significantly lower, so we opted to use BERT base.

5.3 Downstream Extraction Models Results

The training data results show significant precision gain, at the expense of some recall drop, which is not a problem as we highlighted earlier. To assess the impact of the filtering setup on the downstream extraction models themselves, and investigate the role of cosine distance threshold, we train several models using the filtered data. There are several architectures used for the AVE task in literature, with a varying degree of complexity (Zalmout et al., 2021). In this part we opt for the original OpenTag model (Zheng et al., 2018). Table 4 shows the results of the filtered compared to raw data, and Fig-

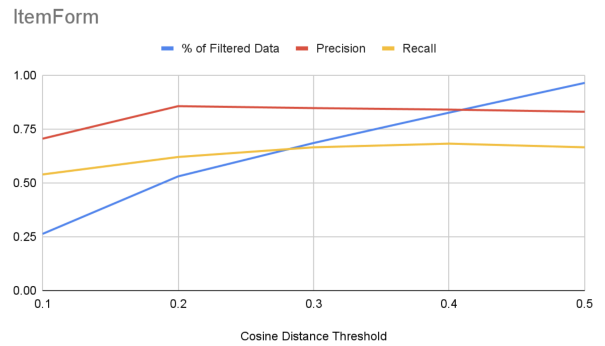


Figure 3: Results for the extraction models trained using the filtered training data for the ItemForm attribute, as a function of the cosine distance threshold.

ure 3 shows the results as a function of the cosine distance threshold. The filtering setup achieves up to 10% absolute gain in precision, with a minimal recall drop of around 1%, after filtering more than 50% of the original training dataset. Interestingly, recall seems to be doing well overall across most of the filtering thresholds, even though we are doing significant filtering of the training data.

5.4 Experimenting with Additional Attributes

We also expanded the experiments to five additional attributes, to further evaluate the consistency of the improvement. We evaluated the resulting training data compared to the unfiltered datasets. Results in Table 5 show gains across all attributes, with an average precision gain of about 9% absolute.

6 Conclusion

We presented an automatic filtering approach using prototype-based representations. We applied the approach on the AVE task, and showed that using centroid-based prototypes outperforms gold-data

Training Data	ContainerType			ItemForm		
	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Raw data	71.3%	34.6%	46.7%	82.4%	67.1%	73.9%
Filtered data*	80.6%	33.5%	47.4%	85.1%	66.8%	74.9%

Table 4: Results for the extraction model evaluation, trained using the filtered data compared to the raw data. *The data is filtered using cosine distance, value-based embeddings, centroid prototype, and the P@R=K setup.

Attribute	Raw Data		Filtered Data	
	Precision	Recall	Precision	Recall
Pattern	74.4%	14.5%	90.8%	14.1%
ItemShape	59.7%	17.8%	70.3%	16.7%
ChocolateType	88.9%	32.9%	93.1%	27.8%
Material	71.6%	16.1%	74.9%	14.5%
ControlType	69.7%	18.0%	80.1%	12.4%
Average	72.9%	19.9%	81.8%	17.1%

Table 5: Training data results for five additional attributes. As explained earlier, recall drop is not as important as precision gain for training data evaluation, and we do not report F1 scores. Check Section 4.4 for more details.

prototypes. We also showed that cosine distance outperforms other outlier detection techniques. We also showed that although recall in the filtered training data drops, the precision gain would still provide the downstream model with the capacity to cover any recall gaps. Model results show significant precision gain, with a minimal drop in recall.

Future work in this direction include tying the filtering process to the underlying task, which would help learn more meaningful representations. Along with developing an iterative filtering process, through which we get the centroids, filter data, then use filtered data to learn centroids again. Such iterative process could improve the quality of the filtering process.

Limitations

Despite the impressive overall performance, along with the simplicity of the approach, the filtering system covers a subset of all possible errors. The goal is to address out-of-context annotations, so errors that are not far off contextually would be more difficult to filter out. Moreover, even for the out-of-context matches, the filtering system is relatively crude and aggressive. The filtering decisions are not fine-grained, so false positives and negatives can still happen. Finally, the centroid prototype, which provides the best results in our setup, is highly dependent on the level of noise in the raw datasets. So we would expect the filtering process to be more biased for attributes that are

excessively noisy. Albeit, we still think the filtering system is powerful, useful, yet simple enough for successful utilization in production.

References

- Li Cui, Deqing Yang, Jiaxin Yu, Chengwei Hu, Jiayang Cheng, Jingjie Yi, and Yanghua Xiao. 2021. Refining sample embeddings with relation prototypes to enhance continual relation extraction. In *Proceedings of ACL-IJCNLP'21 (Volume 1: Long Papers)*, pages 232–243.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. 2018. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Yifan Ding, Yan Liang, Nasser Zalmout, Xian Li, Christan Grant, and Tim Weninger. 2022. Ask-and-verify: Span candidate generation and verification for attribute value extraction. In *Proceedings of EMNLP'22 Industry Track*, Abu Dhabi, UAE.
- Tianyu Gao, Xu Han, Zhiyuan Liu, and Maosong Sun. 2019. Hybrid attention-based prototypical networks for noisy few-shot relation classification. In *Proceedings of AAAI'19*, volume 33, pages 6407–6414.
- Yizhao Gao, Nanyi Fei, Guangzhen Liu, Zhiwu Lu, and Tao Xiang. 2021. Contrastive prototype learning with augmented embeddings for few-shot learning. In *Uncertainty in Artificial Intelligence*, pages 140–150. PMLR.
- Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. [Improving word representations via global context and multiple word prototypes](#). In *Proceedings of ACL'12 (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea.
- Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. [Learning prototype representations across few-shot tasks for event detection](#). In *Proceedings of EMNLP'21*, pages 5270–5277, Online and Punta Cana, Dominican Republic.
- Rongmei Lin, Xiang He, Jie Feng, Nasser Zalmout, Yan Liang, Li Xiong, and Xin Luna Dong. 2021. [Pam: Understanding product images in cross product category attribute extraction](#). In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '21*, page 3262–3270, 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](#).
- Rajendra Pamula, Jatindra Kumar Deka, and Sukumar Nandi. 2011. [An outlier detection method based on clustering](#). In *2011 Second International Conference on Emerging Applications of Information Technology*, pages 253–256.
- Joseph Reisinger and Raymond J. Mooney. 2010. [Multi-prototype vector-space models of word meaning](#). In *Proceedings of NAACL-HLT'10*, pages 109–117, Los Angeles, California.
- Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, volume 30.
- Qifan Wang, Li Yang, Bhargav Kanagal, Sumit Sanghai, D. Sivakumar, Bin Shu, Zac Yu, and Jon Elsas. 2020. [Learning to extract attribute value from product via question answering: A multi-task approach](#). KDD '20, New York, NY, USA. Association for Computing Machinery.
- Xiaochun Wang, Xiali Wang, and Mitch Wilkes. 2021. [A k-nearest neighbor centroid-based outlier detection method](#). In *New Developments in Unsupervised Outlier Detection: Algorithms and Applications*, pages 71–112, Singapore. Springer Singapore.
- Jun Yan, Nasser Zalmout, Yan Liang, Christan Grant, Xiang Ren, and Xin Luna Dong. 2021. [AdaTag: Multi-attribute value extraction from product profiles with adaptive decoding](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4694–4705, Online. Association for Computational Linguistics.
- Li Yang, Qifan Wang, Zac Yu, Anand Kulkarni, Sumit Sanghai, Bin Shu, Jon Elsas, and Bhargav Kanagal. 2022. [Mave: A product dataset for multi-source attribute value extraction](#). In *Proceedings of WSDM '22, WSDM '22*, page 1256–1265, New York, NY, USA.
- Nasser Zalmout, Chenwei Zhang, Xian Li, Yan Liang, and Xin Luna Dong. 2021. [All you need to know to build a product knowledge graph](#). In *Proceedings of KDD'21*, page 4090–4091, New York, NY, USA.
- Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. 2018. [Opentag: Open attribute extraction from product profiles](#). In *Proceedings of KDD'18*.

CGF: Constrained Generation Framework for Query Rewriting in Conversational AI

Jie Hao¹ Yang Liu¹ Xing Fan¹ Saurabh Gupta^{2*} Saleh Soltan¹
Rakesh Chada¹ Pradeep Natarajan¹ Chenlei Guo¹ Gokhan Tur¹

¹Amazon Alexa AI

²LinkedIn

{jieha, yangliud, fanxing, ssoltan, rakchada,
natarap, guochenl, gokhatur}@amazon.com
saurabh3949@gmail.com

Abstract

In conversational AI agents, Query Rewriting (QR) plays a crucial role in reducing user frictions and satisfying their daily demands. User frictions are caused by various reasons, such as errors in the conversational AI system, users' accent or their abridged language. In this work, we present a novel Constrained Generation Framework (CGF) for query rewriting at both global and personalized levels. It is based on the encoder-decoder framework, where the encoder takes the query and its previous dialogue turns as the input to form a context-enhanced representation, and the decoder uses constrained decoding to generate the rewrites based on the pre-defined global or personalized constrained decoding space. Extensive offline and online A/B experiments show that the proposed CGF significantly boosts the query rewriting performance.

1 Introduction

Large-scale conversational AI agents such as Alexa, Siri and Google Assistant help millions of users to perform a lot of tasks, such as playing music, controlling light devices at home, etc. In general, such conversational AI agents have multiple components including automatic speech recognition (ASR) and natural language understanding (NLU). ASR is responsible for converting the speech signal of the user's query (e.g., "play Michael Jackson music") to a text transcript. Following this, NLU provides domain/intent classification (e.g., domain: Music, intent: PlayMusic) and entity labelling (e.g., Artist-Name: Michael Jackson), which are used to fulfill the user's request.

However, sometimes there are frictions due to speech recognition or NLU errors. For example, ASR errors may lead to an erroneous transcript "play alien bridges", when the user actually meant "play leon bridges". Due to such errors, the downstream NLU system is affected, capturing a wrong

entity "alien bridges" for the slot "ArtistName". This leads to a bad user experience and the user has to rephrase their query. Additionally current NLU technology has limitations and cannot handle all the user requests. For example, "tv to input three" cannot be properly handled by NLU (the user's intended request is "turn tv to h.d.m.i. three"). To reduce the friction and make the dialog system more robust, query rewriting (QR) (Ponnusamy et al., 2019; Chen et al., 2020) becomes an increasingly important technique in conversational AI agents. In production conversational AI agents, the QR component is often triggered when the system cannot process user requests with a good confidence. For example, if ASR or the named entity recognition (NER) confidence is low, the QR component can be triggered to automatically map a user query to another form, so that the dialog system can successfully take the right action.

Many existing QR systems use search-based pipelines for either global-wise query rewriting (Fan et al., 2021; Chen et al., 2020) or personalized query rewriting (Cho et al., 2021). These systems typically have two steps: retrieval and ranking. Users' historical defect-free interactions with conversational agents are used to construct the global or personalized index. When a new request arrives, the system compares it to those utterances in the index using a retrieval model such as a dual encoder with billion-scale similarity search (e.g., FAISS) (Johnson et al., 2017) and retrieves top N candidates from the index. Then a ranking model is used to rank these candidates with both neural semantic and IR features as input. The system picks the top 1 ranked candidate as the final rewrite. Such a search-based system is widely used in the large scale conversational AI agents since it can effectively control the output because of the use of the index and thus reduce the risky rewrites.

However, there are also limitations in such retrieval-based systems. First, the query and

*Work done when working at Amazon.

rewrite candidate affinity is mainly captured through a vector dot product and lacks token level modeling. Second, a large memory footprint is needed to store dense representations when a large index is used in the retrieval step.

In this work, we propose to leverage generation-based models under the Constrained Generation Framework (CGF) for the query rewriting task. Since little work has incorporated the previous context information in query rewriting, although its importance is recognized (Wu et al., 2018), we use the previous dialog context and the user’s current request in the encoder. The decoder uses constrained decoding in inference to force the generated rewrite to be in a predefined candidate set. The proposed CGF enables us to mitigate the aforementioned shortcomings from the search-based system since the autoregressive formulation allows the model to directly capture relations between the contextual input and target rewrites and thus effectively cross encode both. Moreover, the memory footprint is greatly reduced because the parameters of our encoder-decoder architecture scale with the vocabulary size, not the index count. Though the neural language generation approaches are known to hallucinate content, our proposed constrained decoding approach with a predefined candidate set makes the generation model faithful to the model input and avoids the potential hallucinations or bad rewrites. We conducted extensive offline experiments for both global and personalized query rewriting to show the effectiveness of the proposed approach. Our online experimental results also demonstrate that the proposed CGF indeed generates rewrites of better quality.

2 Related Work

2.1 Query Rewriting

In dialogue systems, query rewriting benefits dialogue state tracking especially coreference resolution (Rastogi et al., 2019; Vakulenko et al., 2020; Hao et al., 2021), and in general can seamlessly replace the user’s utterance in order to remove friction and unsatisfactory experience to users (Ponnusamy et al., 2019; Wang et al., 2021). To do this, Ponnusamy et al. (2019) proposed to reformulate the queries with a Markov Chain. Chen et al. (2020) proposed a retrieval-based model with a pre-training method. Fan et al. (2021) and Cho et al. (2021) leveraged multi-stage search-based systems to perform global and personalized query

rewriting. In this work, we propose CGF based on Seq2Seq models to generate a rewrite of the initial user query.

Another thread of work that is related to query rewriting is the Grammatical Error Correction (GEC) task. GEC is the task of correcting different kinds of grammatical errors in text such as spelling, punctuation, and word choice errors. Recently, Seq2Seq based models have become the state-of-the-art approach for GEC (Zhao et al., 2019; Wang et al., 2019; Kaneko et al., 2020). The main difference between GEC and our query rewriting is that GEC is more concerned with grammatical corrections, and we focus on the errors from users, ASR or NLU systems to reduce the friction.

2.2 Constrained Generation

Constrained generation has been applied in many tasks such as machine translation and web search. Hokamp and Liu (2017) introduced grid beam search to allow the inclusion of pre-specified lexical constraints. Mohankumar et al. (2021) applied constrained decoding with a diverse sibling search algorithm for search advertising. To the best of our knowledge, ours is the first work that introduces the constrained decoding into query rewriting for conversational AI agents. Moreover, we extend the approach to personalized rewriting to take full advantage of the constrained generation.

3 CGF for Query Rewriting

As shown in Figure 1, we introduce the sequence-to-sequence (Seq2Seq) model to generate the rewrite, where a bidirectional encoder takes the context and current request as input, and an autoregressive decoder relies on the pre-defined index to perform the constrained decoding in order to generate the target rewrite.

3.1 Context-enhanced Modeling

We adopt the Seq2Seq pre-trained model BART (Lewis et al., 2020). It has the same model architecture as the widely-used Transformer model (Vaswani et al., 2017) and is pre-trained with a denoising way (Devlin et al., 2019). In this work, we flatten the previous dialogue turns (including both user requests and agent responses) and the current user request into a single sequence for the encoder input, as shown in Figure 1, and fine-tune BART.

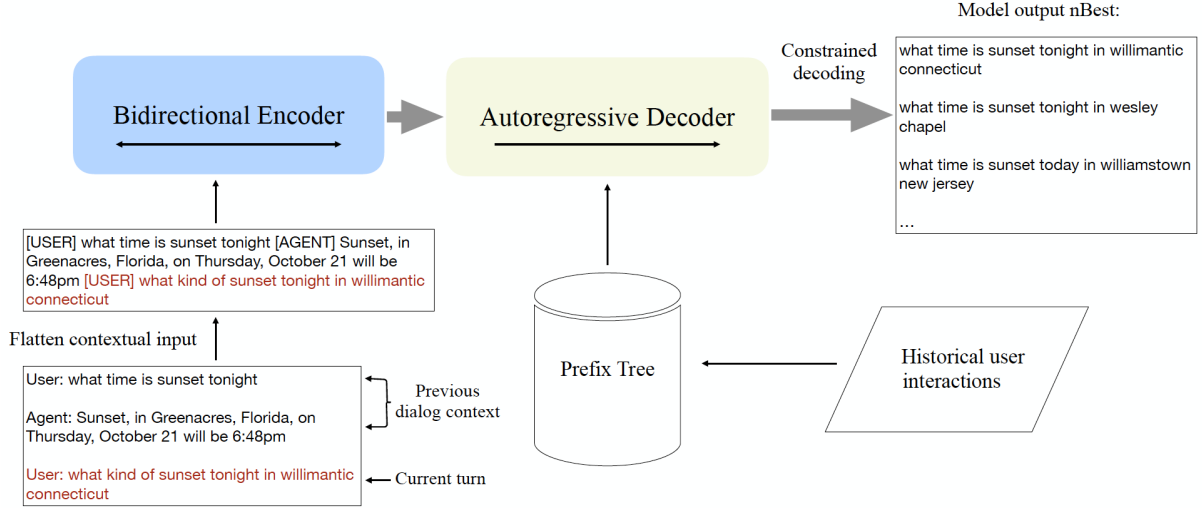


Figure 1: Illustration of the Constrained Generation Framework (CGF) for query rewriting. When a new utterance arrives, the model performs the contextual encoding and constrained decoding and outputs the final rewrites. “Model output nBest” denotes multiple candidates generated using beam search.

Formally, given a context-enhanced request sequence $\mathbf{Q} = \{q_1, \dots, q_M\}$, where q_i denotes a token in the sequence, and the corresponding rewrite $\mathbf{R} = \{r_1, \dots, r_N\}$. The encoder is responsible for reading the input request and its previous dialogue turns, and the decoder autoregressively generates the rewrites. Given the hidden representations of the context-enhanced request and the rewrite, the conditional probability of the n -th target word \mathbf{r}_n is calculated as following:

$$\mathbf{H}_{Enc} = \text{ENC}_{BART}(\mathbf{Q}^0), \quad (1)$$

$$\mathbf{H}_{Dec} = \text{DEC}_{BART}(\mathbf{R}^0, \mathbf{H}_{Enc}) \quad (2)$$

$$\mathbf{P}(\mathbf{r}_n | \mathbf{R}_{<n}, \mathbf{Q}; \theta) = \text{Softmax}(\text{Proj}(\mathbf{h}_n)) \quad (3)$$

where \mathbf{h}_n is the n -th hidden representation of \mathbf{H}_{Dec} . $\text{Proj}()$ and $\text{Softmax}()$ are two transformation functions in the output layer of the decoder (Vaswani et al., 2017).

3.2 Constrained Decoding

Neural language generation approaches are known to hallucinate content, resulting in generated text that conveys information that does not appear in the input. For example, if a user has a request “play *broadway girls*”, the model with free-style generation can generate a rewrite “play *broadway girls by morgan wade*”. This is factually wrong since “morgan wade” never sings the song “broadway girls”. This is because general generative models leverage the beam search over the entire vocabulary and thus there is a chance of generating fluent but

factually incorrect sentences. Thus, the inability to effectively control the generated text has become one of the biggest obstacles for adopting generative models for query rewriting in conversational AI. In this work, we propose to use constrained decoding in the generative models to reduce the potential bad rewrites.

Beam search has been widely used in Seq2Seq models during inference to improve the search quality. The standard beam search consists of selecting the top B hypotheses with the highest log probability $S(\hat{r}_t, \hat{r}_{<t} | Q) = S(\hat{r}_{<t} | Q) + \log P(\hat{r}_t | \hat{r}_{<t}, Q)$ at each time step t , where \hat{r}_t denotes the token in the generated hypothesis. Allowing to generate any token from the vocabulary at every decoding step might lead the model to generate output strings that are not valid (i.e., bad rewrite). Hence, we resort to constrained beam search, forcing to only decode valid rewrites from a predefined candidate set. We define our constraint in terms of a prefix tree T , where nodes are tokens from the vocabulary. For each node $t \in T$, its children indicate all the allowed continuations from the prefix, which is defined as traversing the trie from the root to t . More formally, when decoding the token r_t at time step t , the constrained probability distribution is calculated as:

$$\tilde{P} = \begin{cases} P(\hat{r}_t = r | \hat{r}_{<t}, Q), & \text{if } r \in \text{suffix}_T(\hat{r}_{<t}) \\ 0, & \text{otherwise} \end{cases}$$

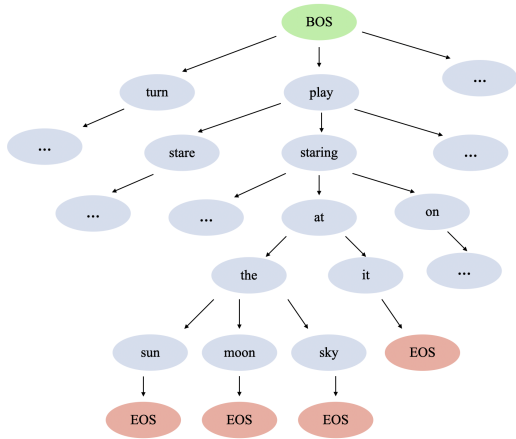


Figure 2: A snapshot of the utterance trie we construct based on the global index. When the model has generated a sequence “[BOS] *play staring at*” during the decoding process, in the next step, using the pre-defined trie, the model is only allowed to generate either “*the*” or “*it*”. Then, if the model generates “*the*” next, it is only allowed to generate one of the three words “*sun*”, “*moon*” or “*sky*” in the step after it.

where we remove all the tokens r that are not a suffix of the already generated sequence $\hat{r}_{<t}$ in the trie. In this way, we can ensure that the model is only allowed to generate the rewrites from the predefined candidates set.¹

In the trie shown in Figure 2, each path from the root node to the leaf node (e.g., [BOS] \rightarrow *play* \rightarrow *staring* \rightarrow *at* \rightarrow *it* \rightarrow [EOS]) represents an utterance that we allow the model to generate. “[BOS]” is the special token indicating the beginning of a sequence. Similarly, “[EOS]” denotes the end of a sequence.

3.3 Global and Personalized Query Rewriting

Constrained generation with the predefined decoding space can not only reduce the risks, but also offer flexibility to conduct rewrite with utterance sets predefined at different granularities. In this section, we introduce how to conduct the global and personalized query rewriting with CGF.

Global Query Rewriting Global query rewriting means that the rewrite for a request is applicable for all the users. For example, for a query “*tv to input three*”, the ideal rewrite for this query is “*turn tv to h. d. m. i. three*”, which is applicable to all the users who might say this request. In the proposed

¹Note that we mask the probabilities of the invalid tokens and do not re-normalize the probability over the vocabulary. We found it is more effective this way.

CGF, we pre-define the global constrained decoding space to include all the rewrite candidates that the model is allowed to generate. To achieve this, inspired by the approach to construct the global index in Fan et al. (2021), we build the global trie that provides rewrite candidates extracted from all the users’ interactions. The global trie is generated from the aggregated, anonymized historical interactions between the users and the agent within a period of time (e.g., 30 days). In addition, after collecting all the user historical interactions, we rely on a defect detection model (Gupta et al., 2021) to filter out the defective utterances. Note that since constrained decoding with the trie doesn’t need to store dense vectors of the index, we can reduce the memory footprint greatly and thus potentially enlarge the trie comparing to the index of search-based models in real online systems.

Personalized Query Rewriting A crucial nature of query rewriting is that often it needs to reflect personal preference or personalized error types to recover from the defect (Cho et al., 2021). For example, for the same defective request “*turn on the moon*”, the intended request for user A may be “*turn on the moonlight sonata*”, whereas user B might want to “*turn on the moon lamp*”. Thus, the global query rewriting described above can not handle such cases. It is necessary to have a personalized query rewriting system to fill this gap. The vanilla Seq2Seq models are not able to perform personalized generation naturally. In contrast, our proposed CGF can allow the generation-based models to perform personalized query rewriting by using a personalized constrained decoding space for each user. For a request coming from a specific user, the model is only allowed to generate a rewrite from the pre-defined personalized decoding space. We follow Cho et al. (2021) to build the constrained decoding space for each user, leveraging their individual interaction history. The utterances included in the constrained decoding space (i.e., trie) reflect satisfied experiences for each user within the past 30 days. In this work, we utilize the model trained with the global training data and apply the personalized trie on it for personalized rewriting.

4 Offline Experiments

4.1 Data

We train our proposed method with weak-labeled data annotated by a model (**Machine-Annotated**). Specifically, we first leverage a defect detection model (Gupta et al., 2021) to find two consecutive deidentified user utterances, where the first turn was defect, but the second turn was successful. Then, we further filter out consecutive utterances with a time gap larger than 35 seconds and edit distance larger than 5. For evaluation, we curated human-annotated test data (**Human-Annotated**). For both global and personalized test sets, we make sure the target rewrites are in the global/personalized constrained decoding space. Table 1 gives the statistics of the data set. Note that all the data has been de-identified.

Data Type	Machine		Human
	Train	Valid	Test
Global QR	6.5m	0.4m	6k
Personalized QR	6.5m	0.4m	5k

Table 1: Statistics of the query rewriting data sets. “Machine” denotes the Machine-Annotated data. “Human” denotes the Human-Annotated data.

4.2 Model Setup

In this work, we fine-tune the pre-trained BART model (Lewis et al., 2020). We compare our proposed model with several baselines. For global query rewriting task, we have two baselines: 1) **DPR** (Karpukhin et al., 2020): we follow a recent retrieval model DPR to train a dual BERT model. 2) **UFS-QR** (Fan et al., 2021): we implement the search-based approach **UFS-QR** that contains a retrieval layer and ranking layer. For personalized query rewriting, we have **Personalized UFS-QR** (Cho et al., 2021) and **DPR** as the baselines. **Personalized UFS-QR** extends the **UFS-QR** by incorporating the personalized features into the ranking model and index construction. In addition, we also compare with the CGF model that uses the global trie. More details of model training from the CGF and baselines training can be found in Appendix A.1. We follow Fan et al. (2021) to build the global trie, which contains 27M unique utterances, and Cho et al. (2021) to build the personalized trie. On memory (disk space) footprint, the global trie we built is 856M, in contrast, the

System	Precision	Trigger Rate
DPR	0.0	0.0
UFS-QR	+10.59%	-13.22%
CGF	+36.62%	+275.23%
<i>Ablations</i>		
CGF w/o CE	+34.43%	+272.34%
CGF w/o CD	+34.97%	+274.02%
CGF w/o both	+32.74%	+266.17%

Table 2: Global query rewriting evaluation. We compare our proposed CGF with the existing search-based query rewriting systems on human annotated test sets. “CE” denotes context-enhanced encoding. “CD” denotes the constrained decoding. All the numbers are relative differences with respect to the baseline: “DPR”.

built FAISS index is 36G for UFS-QR and 89G for DPR with the same utterances.

4.3 Evaluation Metrics

For evaluation, we use utterance level precision and trigger rate. Precision denotes how often the triggered rewrite matches the correct rewrite. The trigger rate is the fraction of instances for which the model makes a prediction with the final beam score above a predefined threshold². We set the threshold to -0.2 for our proposed CGF models.

4.4 Global Query Rewriting Results

Table 2 shows the CGF main results with ablations on the two human-annotated test sets. CGF with context-enhanced encoding and constrained decoding achieves the best performance on precision and trigger rate on the two test sets. Our approach outperforms the search-based UFS-QR system and retrieval system DPR by more than 14% and 21% on precision respectively. Moreover, the proposed approach can confidently trigger more cases.

Table 2 also lists the ablation study results for the global query rewriting task using CGF. “w/o both” denotes the CGF without context-enhanced encoding and constrained decoding, in which the model takes only the query as the encoder input and conduct the unconstrained generation. In particular, although we see that the overall performance of the “w/o CD” model is not bad, it still suffers from hallucinations. Examples with factually incorrect generation can be found in Appendix A.3 Table 5. It is clear that using context-enhanced encoding and

²The final beam score is formalized as $\log(P_\theta(y|x)) = \prod_{i=1}^N p_\theta(y_i|y_{<i}, x)$, where θ is the model parameters and x is the model input.

System	Precision	Trigger Rate
CGF (Global trie)	0.0	0.0
DPR (Personalized index)	+16.03%	-51.85%
Personalized UFS-QR	+16.61%	+15.69%
CGF (Personalized trie)	+19.33%	+17.68%

Table 3: Personalized query rewriting evaluation. All the numbers are relative differences with respect to the baseline: “CGF (Global trie)”.

constrained decoding proved useful. Combining them together is better, resulting in higher precision and at the same time higher trigger rate.

4.5 Personalized Query Rewriting Results

Results for the personalized query rewriting on the Human-Annotated test set using our proposed CGF are in Table 3. We use the same trained model as the global query rewriting task. The only difference is that during inference, the constrained decoding space is changed to the personalized one based on each user’s historical interactions and thus varies across users. As can be seen, CGF outperforms the CGF global model (i.e., Global trie). Also, it outperforms search-based Personalized UFS-QR and DPR respectively by 2.7% and 3.3% on precision, with a higher trigger rate.

5 Online Experiments

5.1 Deployment

In the online system, we run the global and personalized CGF models in parallel. When both the global and personalized components return a rewrite candidate, we prioritize the results from the personalized model over the global one to support any possible personalization of QR. No rewrite will be output from the system if neither model manages to generate a rewrite.

5.2 Online Results

To investigate the effectiveness of the introduced techniques, we leverage the proposed model CGF to generate the rewrites and deploy them into the online environment. We compare it with the no-CGF rewrites within the English speaking users environment. The data was collected for more than one week over a significant percentage of traffic via the A/B testing framework. We use one primary metric to evaluate the performance of our proposed CGF approach during A/B: *Defect Rate*. It denotes the total number of rewritten utterances that are

defective divided by the total number of rewritten utterances. We leverage the defect detection model proposed by Gupta et al. (2021) to measure if an utterance is defective.

From A/B results, we observed significant³ relative reduction of defect rate: **28.97%** and 1 million of new rewrites generated by the proposed approach per week. Table 4 shows the cases where the original requests had unsatisfying responses from the agent and after the rewrite, the friction was removed with satisfying responses. For example, due to an ASR error, the agent response to the original request “how old is tommy in it” cannot fulfill the user’s need. Even without context information, i.e., when the request is the first turn, the CGF can successfully rewrite it, yielding the right response from the agent. More online examples can be found in Table 4.

5.3 Limitations

Trie Coverage Although we have used 27M unique utterances in the global trie and 30 day’s non-defective turns for each user for constrained decoding, the proposed system cannot handle the cold start (e.g., a recent released song) or out-of-trie rewrite cases (the rewrite for the request is not in the trie). To mitigate this, we plan to update the global trie weekly and personalized trie daily. Also we will work on constraining a part of utterance generation (i.e., entity) instead of the entire utterance to enlarge the decoding space.

Latency Generation-based models always suffer from the latency issue due to its autoregressive generation process. In the CGF deployment, we changed the model to ONNX⁴ version and speed up the inference by 30.6%. However, CGF is still 1.5 times slower than the search-based system (Fan et al., 2021). Considering this, we will explore the non-autoregressive approach and related model optimization approaches like distillation, pruning, etc.

6 Conclusion

In this work, we propose CGF, a novel paradigm for query rewriting: generate target rewrite autoregressively with context-enhanced encoding and constrained decoding. CGF is a general framework for different query rewriting purposes where one can

³p-value<0.0001

⁴<https://github.com/onnx/onnx>

Original request (w/o context): USER: how old is tommy in it CGF rewrite: how old is tommyinnit	Agent response (before rewrite): Here’s what I found from the web: A 24-year-old man whose... Agent response (after rewrite): TommyInnit is 17 years old.	Defect Reduction: 50.20%
Original request (w/o context): USER: play night talk by drake CGF rewrite: play knife talk by drake	Agent response (before rewrite): I couldn’t find night talk Agent response (after rewrite): Here’s Knife Talk , by Drake (feat. 21 Savage, Project Pat) , on Amazon Music.	Defect Reduction: 41.52%
Original request (w/ context): USER: play little yancy AGENT: Lil’ Fancy from Apple Music.y USER: play little yankees praise part CGF rewrite: play little yancy praise party	Agent response (before rewrite): Sorry, I’m having trouble finding the song. Agent response (after rewrite): Here is little yancy praise party.	Defect Reduction: 22.71%

Table 4: Online Performance of CGF with Qualitative Examples

freely define the decoding space (e.g., global, personalized or domain-specific space). Both offline and online experiments show that our approach consistently and significantly improves query rewriting performance.

References

- Zheng Chen, Xing Fan, and Yuan Ling. 2020. Pre-training for query rewriting in a spoken language understanding system. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7969–7973. IEEE.
- Eunah Cho, Ziyang Jiang, Jie Hao, Zheng Chen, Saurabh Gupta, Xing Fan, and Chenlei Guo. 2021. Personalized search-based query rewrite system for conversational ai. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 179–188.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.
- Xing Fan, Eunah Cho, Xiaojiang Huang, and Chenlei Guo. 2021. Search based self-learning query rewrite system in conversational ai. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- Saurabh Gupta, Xing Fan, Derek Liu, Benjamin Yao, Yuan Ling, Kun Zhou, Tuan-Hung Pham, and Chenlei Guo. 2021. Robertaiq: An efficient framework for automatic interaction quality estimation of dialogue systems. In *2nd International Workshop on Data-Efficient Machine Learning (DeMaL)*.
- Jie Hao, Linfeng Song, Liwei Wang, Kun Xu, Zhaopeng Tu, and Dong Yu. 2021. RAST: Domain-robust dialogue rewriting as sequence tagging. In *EMNLP*.
- Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. *ACL*.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.
- Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *ACL*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.
- Akash Kumar Mohankumar, Nikit Begwani, and Amit Singh. 2021. Diversity driven query rewriting in search advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3423–3431.
- Pragaash Ponnusamy, Alireza Roshan Ghias, Chenlei Guo, and Ruhi Sarikaya. 2019. Feedback-based self-learning in large-scale conversational ai agents. *arXiv preprint arXiv:1911.02557*.

Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. In *NAACL*.

Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2020. Question rewriting for conversational question answering. *arXiv preprint arXiv:2004.14652*.

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*.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019. Denoising based sequence-to-sequence pre-training for text generation. *arXiv preprint arXiv:1908.08206*.

Zhuoyi Wang, Saurabh Gupta, Jie Hao, Xing Fan, Dingcheng Li, Alexander Hanbo Li, and Chenlei Guo. 2021. Contextual rephrase detection for reducing friction in dialogue systems. In *EMNLP*.

Xianchao Wu, Ander Martinez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. In *NAACL*.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *NAACL*.

A Appendix

A.1 Model Set up

For CGF, we use a batch size of 2048 tokens, dropout rate of 0.1 and adam optimizer. The learning rate is $3E-5$ and linearly warms up over the first 5% steps, then decreases proportionally to the inverse square root of the step number. All the models are trained on eight NVIDIA Tesla V100 GPU.

For the DPR baseline on global rewriting task, we follow [Karpukhin et al. \(2020\)](#) to train the dual BERT model with machine-annotated training set and in-batch negatives. During the inference, for each user request, we use FAISS ([Johnson et al., 2019](#)) search and return top K (K=1 in this work) relevant rewrites from a global index, which contains 27M unique requests as same as global trie. For the personalized rewriting task, similarly, the DPR will return a rewrite from the user’s index, which contains 30 day’s non-defective user historical utterances as same as personalized trie.

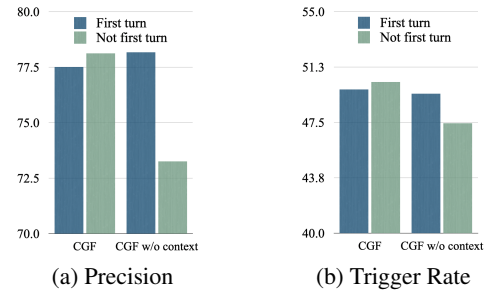


Figure 3: Global query rewriting evaluation on the first turn and not first turn subsets. “CGF w/o context” denotes the CGF without context-enhanced encoding.

A.2 Effect of Context-enhanced Modeling

We study the effect of the context-enhanced modeling in this subsection. As in some of test cases, there are not previous context available and the model will rewrite the first turn of the multi-turn dialogue session. We investigate if the proposed model is robust and effective for both of such cases. Thus, we split the global test set into the with previous context (“First turn”) and without previous context cases (“Not first turn”).

As shown in the Figure 3, the CGF gets significant improvement for both precision and trigger rate on the “Not first turn” test set comparing to CGF without context-enhanced encoding, which demonstrates the effect of the context information during the model training. Moreover, on the “First turn” test set, surprisingly, when there is no previous context for the CGF model, the performance only decreases slightly. This suggests that the model is good at generalization and robust for various test cases in the actual scenario.

A.3 Case Study

We present several representative cases so that we can further understand the effect of the context-enhanced encoding and constrained decoding in CGF. As shown in Table 5, the first example illustrates the cases when CGF w/o context-enhanced encoding gives a rewrite that changes the semantic meanings of the source request (“what kind of” -> “what is”) and is not faithful. However, with consideration of previous context information, the CGF is able to understand the user intent and provide the accurate rewrite. The second case corresponds to the situation of carrying over an correct entity from context and replacing the wrong entity in the current utterance, while as shown in the table, this is not hard for our context-enhanced encoding

Dialog & Reference	CGF	CGF w/o CD	CGF w/o CE
<p>USER: what time is sunset tonight</p> <p>AGENT: sunset, in greenacres, florida, on thursday, october 21 will be 6:48pm</p> <p>USER: what kind of sunset tonight in willimantic connecticut</p> <p>Reference: what time is sunset tonight in willimantic connecticut</p>	<p>what time is sunset tonight in willimantic connecticut</p>	<p>what time is sunset tonight in willimantic connecticut</p>	<p>what is sunset tonight in willimantic connecticut</p>
<p>USER: play little yancy</p> <p>AGENT: Lil' Fancy from Apple Music.y</p> <p>USER: play little yankees praise part</p> <p>Reference: play little yancy praise party</p>	<p>play little yancy praise party</p>	<p>play little yancy praise party</p>	<p>play little yankees praise party music</p>
<p>USER: play in jesus name by katie nicole</p> <p>Reference: play in jesus name by katy nichole</p>	<p>play in jesus name by katy nichole</p>	<p>play in jesus name by kayla nicole</p>	<p>play in jesus name by katy nichole</p>

Table 5: Rewrite examples from offline experiments. In the dialog session, the last turn from the user is the current request which is needed to be rewritten by the model. “CGF w/o CD” denotes the model CGF without constrained decoding, “CGF w/o CE” denotes the CGF without context-enhanced encoding.

models. However, without considering the context information, the model sometimes fails. The third case shows that without constrained decoding, the CGF has a factual inconsistency generation (“kayla nicole” is an artist but never sung “in jesus name”). This is a common situation for generation-based models, especially on unseen data samples. Conversely, this situation rarely happens with constrained decoding, as the generation is based on the predefined constrained decoding space and we will never have such factual inconsistency generation.

Entity-level Sentiment Analysis in Contact Center Telephone Conversations

Xue-Yong Fu, Cheng Chen, Md Tahmid Rahman Laskar,
Shayna Gardiner, Pooja Hiranandani, Shashi Bhushan TN

Dialpad Canada Inc.

Vancouver, BC, Canada

{xue-yong, cchen, tahmid.rahman}@dialpad.com

{sgardiner, phiranandani, sbhushan}@dialpad.com

Abstract

Entity-level sentiment analysis predicts the sentiment about entities mentioned in a given text. It is very useful in a business context to understand user emotions towards certain entities, such as products or companies. In this paper, we demonstrate how we developed an entity-level sentiment analysis system that analyzes English telephone conversation transcripts in contact centers to provide business insight. We present two approaches, one entirely based on the transformer-based DistilBERT model, and another that uses a convolutional neural network supplemented with some heuristic rules.

1 Introduction

Businesses that provide Contact Center as a Service (CCaaS) often leverage Artificial Intelligence (AI) technologies to transcribe telephone conversations and generate aggregated insights reports for contact centers across various industry verticals. Customers occasionally make evaluative comments about specific products or companies during a customer support call to a contact center. These comments provide valuable competitive insights to the business, e.g. positive comments may provide information useful to a marketing department as it formulates an advertising campaign while negative comments may provide valuable insights that can be used to improve a product or a service. In such scenarios, building a system that can identify user sentiments towards entities like product or companies could be useful.

Though Aspect-Based Sentiment Analysis (ABSA) (Zhou et al., 2019, 2020) that aims to extract the user sentiment expressed towards a specific aspect associated with a given target could be a possible solution to solve such problems, it should be noted that ABSA has a key limitation in this regard. For example, in tasks like competitor analysis where the objective is to understand the overall user sentiment within a certain period

of time on specific products or general company services, ABSA techniques could not be useful as they provide more fine-grained opinions towards predefined aspects (e.g. features of a product, ease of use of a software, or aspects of restaurant experience) instead of providing a generic user sentiment towards a specific entity (Zhou et al., 2019).

While building the Entity-level Sentiment Analysis (ELSA) system for real-world contact center use-cases, we observe several key challenges. First of all, to the best of our knowledge, there are no existing public datasets available for the entity-level sentiment analysis task. Meanwhile, this task becomes more challenging when the requirement is to use a dataset constructing from telephone conversations since constructing a dataset from speech transcripts generated from telephone conversations is non-trivial as telephone transcripts are generated by automatic speech recognition (ASR) systems that have their own unique characteristics. For instance, an ASR system may have mistranscription errors as well as linguistic disfluencies (e.g. filler words) (Fu et al., 2021) that usually occur in a conversational speech dataset (Malik et al., 2021).

The factors mentioned above make the implementation of an entity-level sentiment analysis model very challenging to detect user opinions towards entities that appear in contact center calls. In this paper, we address the existing limitations behind developing an entity-sentiment model for commercial scenarios in the domain of business telephone conversation data in contact centers. Since there is no suitable publicly available dataset for the entity-level sentiment analysis task, we briefly describe how we sampled and annotated the data for this task. We then propose two approaches that leverage neural models for this task (i) one is based on the DistilBERT (Sanh et al., 2019) model in which we modify its architecture such that the model can also be utilized to extract the opinion term(s) while detecting the sentiment po-

larity towards a named entity; (ii) while the other approach uses a convolutional neural network (dos Santos and Gatti, 2014) supplemented with some pre-defined heuristic rules. We compare the effectiveness of both approaches through extensive experiments and discuss our findings to provide valuable insights for future developments of ELSA models for real world commercial scenarios.

2 Related Work

Since the entity-level sentiment analysis task is closely related to aspect-level sentiment analysis, in this section, we first briefly review the aspect-level sentiment analysis task followed by the entity-level sentiment analysis task in order to clarify the distinction between these two tasks while discussing our rationale behind developing an entity-level sentiment analysis model for contact centers.

2.1 Aspect-Based Sentiment Analysis (ABSA)

ABSA aims to classify the sentiment polarity of aspects of certain objects. Many previous studies are focused on this research (Sun et al., 2019; Tang et al., 2016; He et al., 2018; Zhao et al., 2020; Zhou et al., 2019). A more fine-grained related task is aspect sentiment triplet extraction (ASTE) (Peng et al., 2020; Xu et al., 2020), which extracts a triplet – aspect term, opinion term and sentiment – from the input. Detection of aspects in both ABSA or ASTE often relies on implicit lexical or semantic signs, for instance, *the food is too spicy* suggests that this comment is about the *taste* aspect. This is different from the entity recognition task where the goal is to detect the named entities in a given utterance based on the overall context.

2.2 Entity-level Sentiment Analysis (ELSA)

ELSA aims to predict the sentiment of named entities in a given text input (Steinberger et al., 2011; Saif et al., 2014). These named entities are usually application dependent. One recent work on ELSA is the work of Luo and Mu (2022), where they studied entity sentiment in news documents. Another prominent work on ELSA is the work of Ding et al. (2018), where an entity-level sentiment analysis tool was proposed for Github issue comments. Contrary to the above studies that focused on typed text, our focus is on noisy textual data (i.e., speech transcripts). Moreover, our proposed models can infer both entity sentiment and corresponding opinion terms for a better analysis of user

sentiments towards products or companies in business telephone conversations in contact centers.

3 Task Description

Let us assume that we have an utterance $U = w_1, w_2, \dots, w_n$ containing n words. The goal of the ELSA task is to identify m opinion words $O_W = ow_1, ow_2, \dots, ow_m$, (where $m < n$), and classify the sentiment of the identified opinion words towards the target entity e in the given utterance. In Table 1, we show some examples of the ELSA task to detect user sentiments towards products and organization type entities. In the first two examples, the customer is directly expressing positive sentiment about the named entity. For instance, (i) they say “I love it” indicating “Google” in context, or (ii) they are “very impressed” with “MAC”. In the third and fourth examples, customers are expressing negative sentiment about a product or facet associated with the company, e.g., “He has a hard time finding a good yogurt from Walmart” is a comment about the quality of Walmart’s service, not a comment about yogurt. Similarly, in the fourth example, difficulty navigating the Instacart app is indirectly an indication of negative sentiment concerning Instacart.

4 Dataset Construction

As noted earlier, there is no publicly available dataset for the ELSA task. We therefore had to create and annotate our own dataset. The first major issue that we observed while constructing a dataset for ELSA is that the entity-level sentiment events in our telephone transcripts are very infrequent. Hence, random data sampling techniques might yield an imbalanced dataset where most utterances would not have any positive or negative sentiments towards an entity. We therefore used two pre-existing models — a named entity recognition (NER) model based on DistilBERT (Sanh et al., 2019) that was trained to identify **Organization** and **Product** type entities and a convolutional neural network (CNN) (Krizhevsky et al., 2012; dos Santos and Gatti, 2014; Albawi et al., 2017) sentiment analysis model — to sample 13000 utterances that contained at least one named entity and one positive or negative sentiment predicted by these models. To balance the dataset, we sampled an additional 10000 utterances containing at least one entity and having no polarized sentiments (i.e., only neutral sentiment). The resulting 23000

I work at Google and I love it a lot.
She’s very impressed how MAC works so well.
He has hard time finding a good yogurt from Walmart .
It’s quite difficult to navigate the mobile app of Instacart .

Table 1: Examples for entity-level sentiment analysis. Words in color *blue* are target named entities. Words in color *teal* are positive opinion words. Words in color *purple* are negative opinion words.

utterances were manually annotated by independent annotators to determine the positive, neutral, or negative sentiment toward the target entity. The annotators also identified the opinion terms in the utterances.

5 Our Proposed Models

For performance evaluation, we propose two approaches: (i) *DistilBERT-based Model*, and (ii) *CNN-based Model with Heuristics Rules*. Below, we present our proposed approaches.

5.1 DistilBERT-based Model

For this approach, we leverage the DistilBERT model since this is a very lightweight model that does not require much computing power in production environments (Sanh et al., 2019). Below, we describe how we utilize this model for ELSA.

NER tagging: Given an utterance as input, we first run an NER model to determine if there is at least one entity (product or organization) detected. Our NER model is based on DistilBERT that is trained over business conversation data collected from call centers. During the training stage, we use the cross entropy (CE) loss as defined in Equation 1:

$$L_{CE} = -\frac{1}{N} \sum_{n=1}^N \log \frac{e^{\hat{y}_{n,y_n}}}{\sum_{c=1}^C e^{\hat{y}_{n,c}}} \quad (1)$$

Here, N is the number of samples in a batch, and C denotes the number of classes, $\hat{y}_{n,c}$ is the logit of the c -th class in the n -th example, and \hat{y}_{n,y_n} is the logit of the gold class in the n -th example.

Context Representation: We insert a special tag, *_NE_*, before any named entities detected in an utterance. This helps the model to identify which spans belong to the entity. For example, if the raw input is “*I really don’t like using Snapchat*”, we reformulated the input as “*I really don’t like using _NE_ Snapchat*”. Then, we send the pre-processed input to our entity sentiment detection model that we describe below.

Entity Sentiment Detection: Our entity sentiment detection model is also based on DistilBERT. However, for this task, we train DistilBERT over business telephone conversation data for a different task: the sentiment classification task. Meanwhile, our entity sentiment detection model can also extract the opinion word(s) in a given utterance. This is done by adding an additional prediction layer on top of the DistilBERT model to identify the opinion words. During the training phase, the model is fine-tuned on our entity sentiment dataset to predict the polarity of the opinion terms for a given utterance. If the target entity’s sentiment is positive or negative, the model will assign respective tags (**POS** for positive and **NEG** for negative) to the opinion token(s), while the remaining tokens will be assigned to the **O** tag.

Transfer Learning: To improve model performance, we introduce a transfer learning technique for our entity sentiment detection model, for which we first fine-tune the DistilBERT model for the sentence classification task (i.e., sentiment analysis) on the Stanford Sentiment Treebank (SST) dataset that contains 11,855 training examples. The assumption is that if a model is fine-tuned on a similar task, it is expected to perform better on related downstream tasks (Laskar et al., 2022c; Garg et al., 2020). Although the SST dataset is about predicting the general sentiment of a given text sequence, it requires the model to learn what words are associated with positive sentiments and what are associated with negative sentiment, which is essential for our task. The DistilBERT model trained on the SST dataset is then fine-tuned again on the processed input (pre-processed by using the *_NE_* tag obtained from our NER model) in the contact center conversation dataset. As mentioned earlier, in this stage of fine-tuning, the entity sentiment model can also extract the opinion word(s) via utilizing the additional prediction layer that we added on top of DistilBERT.

An overview of our proposed DistilBERT-based approach is shown in Figure 1.

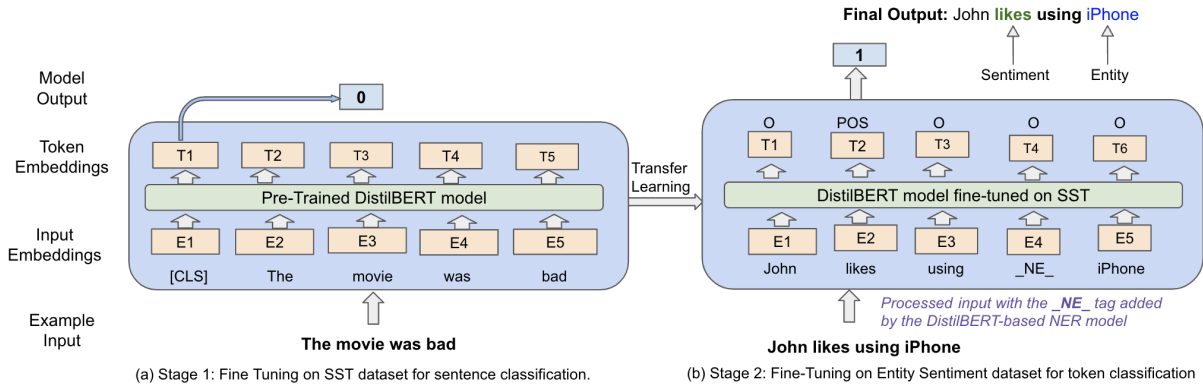


Figure 1: An overview of our proposed DistilBERT-based approach: (a) first, we do fine-tuning on the SST dataset for the generic sentiment analysis task, and (b) then, fine-tune on the in-domain Entity Sentiment dataset for entity level opinion extraction. Here, in the output layer, 1 denotes positive while 0 denotes negative sentiment.

5.2 CNN-based Sentiment Model Supplemented with Heuristic Rules

For this model, we employ a two-step approach. We first run a general sentiment analysis model that classifies the sentiment of a given utterance and also extracts the keywords that cause that sentiment (if the sentiment is positive or negative). We treat these sentiment keywords as opinion word candidates. Then we employ a set of linguistic heuristics that identify the opinion words that are associated with the entities mentioned in the input.

The sentiment analysis model is a multiclass, CNN-based classification model. We choose CNN here due to its effectiveness in related tasks (e.g., ABSA task) (Wang et al., 2021). However, for ELSA, we add an explainability layer on top of CNN that is tasked with explaining predictions. The explainability technique that we leverage is called Integrated Gradients, adopted from (Sundararajan et al., 2017). After a sentiment score is predicted by the model, the explainability layer emits words that are highly associated with the predicted sentiment. Note that we apply some heuristics to select the emitted words as candidates for opinion words.

Heuristics: For the extraction of opinion words, we utilize heuristics based on phrase structure types that are most likely to contain entity sentiment. We divide these phrase types into three categories to find which part of speech contains the potential opinion word: verb-based, adjective-based, and noun-based. Table 2 illustrates the possible syntactic patterns captured by these heuristics, all of which also allow for optional modifiers such as intensifiers (e.g. *really*, *very*, *so*), complementizers

(*that*, *which*) or stacked adjectives. Some example phrases that were captured include: *I'm so happy that Google made this*, *Android sucks*, *that was awesome of Netflix to do*, *Netflix is garbage*, *my hatred of LaTeX*, *classic LaTeX awesomeness*, etc.

6 Experiment

In this section, we present the training parameters, evaluation metrics, and the experimental results. In our experiments, we use the following three models:

- **DistilBERT:** This model does not leverage the SST dataset as the first stage of fine-tuning. Instead, it is fine-tuned only on our ELSA dataset.
- **DistilBERT + SST:** This model is initially fine-tuned on the SST dataset and then fine-tuned on our ELSA dataset.
- **CNN + Heuristics:** This is the model that leverages CNN and supplemented with some heuristics rules.

6.1 Training Parameters

For the DistilBERT model, we set the batch size to 32, learning rate to 5×10^{-5} , and employ early stopping with patience set to 5. The pretrained model is based on the HuggingFace Transformer (Wolf et al., 2020). While for the CNN model, we use 300 dimensional fastText embeddings (Bojanowski et al., 2016; Santos et al., 2017), global max pooling is utilized in the convolutional layer with filter sizes: 2, 3, 4, 5, 6. The fully connected layer is 128 dimensional.

Verb-Based	Adjective-Based	Noun-Based
sentiment verb + entity	sentiment adjective + entity	entity + sentiment noun
sentiment verb + + preposition + entity	sentiment adjective + <i>of/for</i> + entity	sentiment noun preposition + entity
entity + sentiment verb		entity + aux verb sentiment noun

Table 2: Heuristic Rules to Extract Opinion Words

Models	Precision		Recall		F1	
	Ent	OP	Ent	Op	Ent	Op
DistilBERT	74.43	59.99	73.77	69.44	73.70	64.35
DistilBERT + SST	74.83	68.21	74.69	63.02	74.72	65.48
CNN + Heuristics	77.48	97.65	58.23	16.78	50.07	28.64

Table 3: Experimental results for the Entity Level Sentiment (Ent) and Opinion Word Extraction (OP) tasks.

6.2 Evaluation Metrics

To evaluate entity sentiment classification and opinion word extraction, we define two kinds of evaluation metric. For polarity classification, we calculate precision, recall and F1 score for three sentiment categories: positive, negative and neutral. We then calculate the weighted value of these (support-based). For opinion word extraction, we evaluate it using the metrics that are usually used for named entity recognition (Li et al., 2020) and calculate precision, recall and F1 score for opinion words. The evaluation was done in a sample of 175 annotated utterances that were reviewed by another group of annotators.

6.3 Results

From Table 3, we find that in terms of the F1 metric, both variations of DistilBERT – (Vanilla **DistilBERT** and **DistilBERT + SST**) – outperform the **CNN + Heuristics** model by a huge margin. More specifically, **DistilBERT + SST** outperforms the **CNN + Heuristics** model by 15.75% of the F1 score. Comparing **DistilBERT** and **DistilBERT + SST**, we can see the effect of SST pre-training, which brings the F1 score up from 73.7% to 74.72%, with an increase of 1.38%. We also find that the **CNN + Heuristics** model obtains impressive precision score. This is because the heuristic rules used in the **CNN + Heuristics** model were developed to emphasize precision, but they do not handle linguistic variation well, resulting in poor Recall and F1 scores.

For opinion word extraction, which is noted as **OP**, the performance gap between the **DistilBERT** model and the **CNN + Heuristics** model is even larger. As shown in Table 3, **DistilBERT + SST**

outperforms the **CNN + Heuristics** by 38.48% F1 score. This is mainly because the **CNN + Heuristics** has very poor performance in recall: only 16%. Although the recall of **DistilBERT + SST** is lower than **DistilBERT**, its F1 score is still 1.07% higher than its counterpart.

Robustness Test: The overall metrics can’t identify if the performance of a model is robust in different situations. Thus, we investigate if our proposed model is robust against various kinds of input texts. For this purpose, we separate the test data into different sub-populations by the number of tokens and the number of entities. We then evaluate our models on sub-populations to see how they perform. Below, we define these sub-populations.

- (i) **1 entity:** *input text has only one target entity.*
- (ii) **> 1 entity:** *input text has more than one target entity.*
- (iii) **< 8 tokens:** *input text with less than eight tokens.*
- (iv) **> 45 tokens:** *input with more than forty five tokens.*

Table 4 contains the results of our proposed models in different data slices. We find that both models perform poorly when the input is long (> 46 tokens) compared to when the input is short (< 8 tokens). This could be because it is much harder to model long term dependencies when the sequence length is too long.

We also find that the **DistilBERT + SST** model is more sensitive to the number of target entities in the input compared to the **CNN + Heuristics** model. Its F1 score drops by 7.16% when the number of target entities increases from one to more than one.

Model	Precision	Recall	F1
CNN + Heuristics	97.65	16.78	28.64
CNN + Heuristics (< 8 tokens)	100	38.89	56
CNN + Heuristics (> 45 tokens)	100	8.16	14.99
CNN + Heuristics (= 1 entity)	97.54	16.86	28.75
CNN + Heuristics (> 1 entity)	100	15.56	26.72
DistilBERT + SST	68.21	63.02	65.48
DistilBERT + SST (< 8 tokens)	66.9	79.46	72.46
DistilBERT + SST (> 45 tokens)	62.18	26.32	36.92
DistilBERT + SST (= 1 entity)	68.28	63.67	65.87
DistilBERT + SST (> 1 entity)	68	52.94	58.71

Table 4: Robustness Report on the Opinion Word Extraction task.

7 Commercial Application

We have deployed the **DistilBERT + SST** model in our production system to generate entity sentiment data for contact centers as it has better accuracy than the **CNN + Heuristics** model. Due to the small model size and efficient inference of DistilBERT, each model instance is assigned 1 CPU and 1GB memory. Once there are enough entity-level sentiment predictions, there is a dedicated pipeline to aggregate entity sentiment in different granularity for each customer.

There are many use cases of the aggregated insights of entity sentiment. Contact center managers can use this information to improve contact center efficiency by investigating why customers are not happy with certain products (e.g., *itelephone 13 Pro Max*) and develop desired responses when the customer is complaining about it, so that the agents can handle the difficult situation more efficiently. The collected negative feedback can be used to inform the product team how to improve the products. The insights can also be used to conduct comparisons between several products or companies and help with competitor analysis.

8 Conclusion

In this paper, we described the creation of a task-specific dataset and a new model that extracts opinion words while performing entity sentiment polarity detection. The resulting DistilBERT-based model is currently deployed as a commercial application for entity-level sentiment analysis for English contact center conversations. In the future, we will investigate how to extend our proposed methods to other applications (Laskar et al., 2022a,b) of the entity recognition task (Fu et al., 2022) in telephone transcripts and explore how to improve model performance on utterances that contain more than one entity.

Limitations

As our entity sentiment models are trained on English business telephone conversations, they might not be suitable to be used in other domains, types of inputs (i.e written text), or languages. The NER component of **DistilBERT** based model has some limitations while detecting product and organization type entities. It is more biased towards detecting the entities that appear more frequently in the training data and misses rare entities. This could impact the overall performance of the model.

Ethics Statement

This data in this research is comprised of individual sentences that do not contain sensitive, personal, or identifying information. The entity sentiment model deployed in production is not used to attach any sentiment to people, only to non-human entities. Each machine-sampled utterance is labelled by annotators before the utterance is used as part of the training dataset. While annotator demographics are unknown and therefore may introduce potential bias in the labelled dataset, the annotators are required to pass a screening test before completing any labels used in these experiments, thereby mitigating this unknown to some extent. We paid adequate compensation to the annotators. Future work should nonetheless strive to improve training data further in this regard.

ACKNOWLEDGEMENTS

We would like to thank Simon Corston-Oliver for his helpful and detailed feedback on the paper. We also thank the reviewers for their excellent review comments that helped us improving the paper.

References

- Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. arxiv 2016. *arXiv preprint arXiv:1607.04606*.
- Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering, SEmotion@ICSE 2018, Gothenburg, Sweden, June 2, 2018*, pages 7–13. ACM.
- Cícero Nogueira dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 69–78. ACL.
- Xue-Yong Fu, Cheng Chen, Md Tahmid Rahman Laskar, Shashi Bhushan, and Simon Corston-Oliver. 2021. Improving punctuation restoration for speech transcripts via external data. In *Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021)*, pages 168–174.
- Xue-Yong Fu, Cheng Chen, Md Tahmid Rahman Laskar, Shashi Bhushan Tn, and Simon Corston-Oliver. 2022. An effective, performant named entity recognition system for noisy business telephone conversation transcripts. In *Proceedings of the Eighth Workshop on Noisy User-generated Text (W-NUT 2022)*, pages 96–100.
- Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7780–7788.
- Ruidan He, Wee Sun Lee, Hwee Tou Ng, and Daniel Dahlmeier. 2018. Effective attention modeling for aspect-level sentiment classification. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1121–1131. Association for Computational Linguistics.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114.
- Md Tahmid Rahman Laskar, Cheng Chen, Jonathan Johnston, Xue-Yong Fu, Shashi Bhushan TN, and Simon Corston-Oliver. 2022a. An auto encoder-based dimensionality reduction technique for efficient entity linking in business phone conversations. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3363–3367.
- Md Tahmid Rahman Laskar, Cheng Chen, Aliaksandr Martsinovich, Jonathan Johnston, Xue-Yong Fu, Shashi Bhushan Tn, and Simon Corston-Oliver. 2022b. BLINK with Elasticsearch for efficient entity linking in business conversations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 344–352, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.
- Md Tahmid Rahman Laskar, Enamul Hoque, and Jimmy Xiangji Huang. 2022c. Domain adaptation with pre-trained transformers for query-focused abstractive text summarization. *Computational Linguistics*, 48(2):279–320.
- Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70.
- Manman Luo and Xiangming Mu. 2022. Entity sentiment analysis in the news: A case study based on negative sentiment smoothing model (nssm). *International Journal of Information Management Data Insights*, 2(1):100060.
- Mishaim Malik, Muhammad Kamran Malik, Khawar Mehmood, and Imran Makhdoom. 2021. Automatic speech recognition: a survey. *Multimedia Tools and Applications*, 80(6):9411–9457.
- Haiyun Peng, Lu Xu, Lidong Bing, Fei Huang, Wei Lu, and Luo Si. 2020. Knowing what, how and why: A near complete solution for aspect-based sentiment analysis. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8600–8607. AAAI Press.
- Hassan Saif, Yulan He, Miriam Fernandez, and Harith Alani. 2014. Semantic patterns for sentiment analysis of twitter. In *International Semantic Web Conference*, pages 324–340. Springer.
- 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.
- Igor Santos, Nadia Nedjah, and Luiza de Macedo Mourelle. 2017. Sentiment analysis using convolutional neural network with fasttext embeddings. In *2017 IEEE Latin American conference on computational intelligence (LA-CCI)*, pages 1–5. IEEE.

- Josef Steinberger, Polina Lenkova, Mijail Kabadjov, Ralf Steinberger, and Erik Van der Goot. 2011. Multilingual entity-centered sentiment analysis evaluated by parallel corpora. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 770–775.
- Chi Sun, Luyao Huang, and Xipeng Qiu. 2019. Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. 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 380–385, Minneapolis, Minnesota. Association for Computational Linguistics.
- Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.
- Duyu Tang, Bing Qin, and Ting Liu. 2016. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 214–224. The Association for Computational Linguistics.
- Xinyi Wang, Feng Li, Zequn Zhang, Guangluan Xu, Jingyuan Zhang, and Xian Sun. 2021. A unified position-aware convolutional neural network for aspect based sentiment analysis. *Neurocomputing*, 450:91–103.
- 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 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.
- Lu Xu, Hao Li, Wei Lu, and Lidong Bing. 2020. Position-aware tagging for aspect sentiment triplet extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2339–2349. Association for Computational Linguistics.
- Fei Zhao, Zhen Wu, and Xinyu Dai. 2020. Attention transfer network for aspect-level sentiment classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 811–821, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Jie Zhou, Jimmy Xiangji Huang, Qin Chen, Qinmin Vivian Hu, Tingting Wang, and Liang He. 2019. Deep learning for aspect-level sentiment classification: survey, vision, and challenges. *IEEE access*, 7:78454–78483.
- Jie Zhou, Jimmy Xiangji Huang, Qinmin Vivian Hu, and Liang He. 2020. SK-GCN: modeling syntax and knowledge via graph convolutional network for aspect-level sentiment classification. *Knowl. Based Syst.*, 205:106292.

QUILL: Query Intent with Large Language Models using Retrieval Augmentation and Multi-stage Distillation

Krishna Srinivasan* Google Research krishnaps@google.com	Karthik Raman* Google Research karthikraman@google.com	Anupam Samanta Google anupamsamanta@google.com
Lingrui Liao Google lingrui@google.com	Luca Bertelli Google lbartelli@google.com	Mike Bendersky Google Research bemike@google.com

Abstract

Large Language Models (LLMs) have shown impressive results on a variety of text understanding tasks. Search queries though pose a unique challenge, given their short-length and lack of nuance or context. Complicated feature engineering efforts do not always lead to downstream improvements as their performance benefits may be offset by increased complexity of knowledge distillation. Thus, in this paper we make the following contributions: (1) We demonstrate that Retrieval Augmentation of queries provides LLMs with valuable additional context enabling improved understanding. While Retrieval Augmentation typically increases latency of LMs (thus hurting distillation efficacy), (2) we provide a practical and effective way of distilling Retrieval Augmentation LLMs. Specifically, we use a novel two-stage distillation approach that allows us to carry over the gains of retrieval augmentation, without suffering the increased compute typically associated with it. (3) We demonstrate the benefits of the proposed approach (QUILL) on a billion-scale, real-world query understanding system resulting in huge gains. Via extensive experiments, including on public benchmarks, we believe this work offers a recipe for practical use of retrieval-augmented query understanding.

1 Introduction

The recent advent of billion+ parameter Large Language Models (LLMs) – such as T5 (Raffel et al., 2019), mT5 (Xue et al., 2021), GPT-3 (Brown et al., 2020) and most recently PaLM (Chowdhery et al., 2022) – has disrupted many language understanding tasks – with new benchmarks set or eclipsed routinely by these Transformer models and their variants.

Queries – especially keyword search ones – present a unique challenge though. Their short

length, inherent ambiguity and lack of grammar mean query understanding tasks typically require more memorization and world knowledge than other NLP tasks (Broder et al., 2007). Consequently, despite LLMs leading performance on language and query understanding tasks – like intent classification, query parsing and relevance prediction – there is significant room for further improvement.

In this paper we leverage Retrieval-Augmentation to provide LLMs more context and grounding for search queries. We show that the titles and URLs of documents retrieved for the query, greatly help improve LLMs query understanding capabilities. While different retrieval augmentation models exist, we show that even simple concatenation of these titles / urls with the query can help improve LLM performance considerably.

However, the use of retrieval augmentation leads to a new challenge: Increased complexity of LLM inference. More specifically, the quadratic complexity of self-attention in Transformer models means that the latency of LLMs blows up given these (often 10x+) longer input sequences. This presents a significant problem as LLMs are impractical for online use and thus need to be distilled into smaller, more efficient models to be served online. However knowledge distillation (Gou et al., 2021) into these *student* models requires a lot of distillation data annotated by these LLMs – which may not be feasible for these retrieval augmented models.

Thus as a remedy we introduce a new two-stage distillation approach. In the first stage of this approach we distill the retrieval-augmented (long input) LLM (the *Professor*) into a non-retrieval augmented (short input) LLM (the *Teacher*) using a small distillation set. This second LLM *Teacher* is in turn distilled into the final *Student* using a large set.

* Corresponding Authors

Via extensive experiments on a large-scale, real-world problem and data we demonstrate that the resulting QUILL system provides for an efficient and effective way of retaining the performance gains of retrieval augmented LLMs on query understanding tasks.

2 Related Work

Large language models (LLMs) such as mT5 (Xue et al., 2021) demonstrated significant performance improvements on a variety of natural language understanding (NLU) tasks. Specifically in the context of query understanding, researchers found that (a) model size significantly effects the quality of the resulting models (Nogueira et al., 2019; Han et al., 2020), and (b) using additional context in the form of query-associated documents is crucial to the model performance due to the paucity of context available in the query itself (Nogueira and Lin, 2019; Zhang et al., 2020). Retrieval augmentation of the query with the search results retrieved by it is a proven way to incorporate such context in LLM training for NLU tasks, as has been shown recently by models such as RAG (Lewis et al., 2020), REALM (Guu et al., 2020), and RETRO (Borgeaud et al., 2022).

In this paper, we leverage this insight to improve performance of query intent prediction (Broder, 2002) — a crucial query understanding task that is at the heart of modern search engines – using LLMs. Prior work by Broder et al. (2007) found importance of retrieval augmentation using statistical methods for this task. Statistical retrieval augmentation has also been found critical for other query understanding tasks including query expansion (Broder et al., 2008; Diaz and Metzler, 2006) and query tagging (Wang, 2020). We demonstrate similar benefits when using retrieval-augmented LLMs as well.

We also leverage Knowledge Distillation (Hinton et al., 2015; Mirzadeh et al., 2020; Gou et al., 2021) techniques to create a Student model that retains the LLMs gains.

3 Query Intent Understanding

While the techniques described in this paper could be applied to any query understanding task, for the sake of brevity we focus on the task of query intent classification. Query intent (QI) classification is a classical IR task studied for over two decades (Kang and Kim, 2003; Baeza-Yates et al.,

Data	Train	Val	Test	Unlabeled
Orcas-I	1.28M	1K	1K	10.3M
EComm	36K	4K	4K	128M

Table 1: Statistics of datasets used.

2006; Jansen et al., 2008; Kathuria et al., 2010; Lewandowski et al., 2012; Figueroa, 2015; Mohaseb et al., 2019). This task is particularly important in practice, as it is at the top of the search funnel, and the entire search engine behavior may vary based on the predicted query intent. Given the centrality of this task on overall retrieval, models for this task need to be both fast (*i.e.*, low latency) and high efficacy. Thus even a single percentage point quality gain on the QI task can be considered a major accomplishment.

In this paper we tackle the QI task using LLMs. In particular we use two datasets in our study whose details are provided in Table 1:

- **EComm:** Our main dataset will be a **real-world** dataset. Cast as a binary classification problem, this task involves identifying queries with a specific intent – where the required intent is similar to the *transactional* intent of the Broder taxonomy (Broder, 2002) in the context of e-commerce. As common in real-world applications, the human labeled data is accompanied by a large unlabeled set – that is used for knowledge distillation.
- **Orcas-I:** The largest publicly available query intent dataset is ORCAS-I (Alexander et al., 2022). This comprises queries of the ORCAS dataset (Craswell et al., 2020) labeled with one of 5 intent classes. Note that while the test set is human-labeled, the training set labels are weak labels as detailed in ORCAS-I (Alexander et al., 2022) paper’s Methodology section.

4 QUILL Methodology

The keyword nature of queries and lack of context make the QI task (like other query understanding tasks) challenging for LLMs. Thus we propose QUILL as a solution. As seen in Figure 1, QUILL consists of two stages: (a) Retrieval Augmented LLM training, (b) Multi-Stage Distillation into efficient student.

Retrieval Augmented (RA) LLM: The key insight here is that titles / urls of related documents

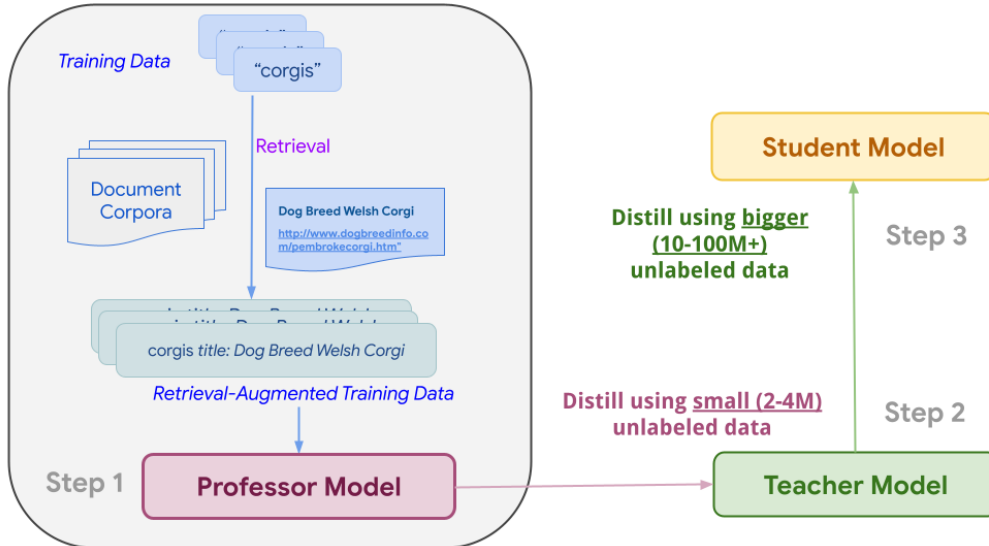


Figure 1: QUILL Architecture : Retrieval Augmentation and Multi-stage Distillation.

Feature	EComm		Orcas-I	
	Median	99%	Median	99%
Query	5	18	5	10
ExpandTerms	17	28	N/A	N/A
(Up to) 10 Titles	157	245	13	104
(Up to) 10 URLs	159	304	39	238

Table 2: mT5 sequence lengths by features.

could provide valuable context to help understand the intent of the query. For example, it may not be immediately apparent what a query like ua 1234 may mean. However, via the retrieved documents we can understand that the query is seeking information about a United Airlines flight.

While there are multiple ways of augmenting the input via retrieved documents (example: the Fusion-in-Decoder architecture (Izcard and Grave, 2020)), we chose to study the most straightforward and popular approach of concatenating the titles / urls of the retrieved top-k documents with the original query as the input to our LLM. As shown empirically (Sec 5), this model outperforms all baselines – demonstrating the value of additional context.

Multi-stage distillation: The drawback of RA is the additional sequence length of the input. As seen from Table 2, augmenting a query with (upto) 10 titles and urls increases the sequence length by an order of magnitude. Consequently, this makes distillation far more challenging given the quadratic complexity of sequence length (due to self-attention) in transformer models. This leaves

us in a dichotomy between a more effective model with a much smaller distillation set, vs. a lower performing model with a larger distillation set. Given a large distillation set is required for training an effective student, this leaves us at risk of not being able to benefit from RA, given that a very large dataset with RA will incur very long and impractical inference times.

To get the best of both worlds we propose a two-stage distillation approach. In the first stage we distill the *Professor* RA LLM into a *Teacher* LLM without RA. The Teacher model uses ExpandTerms which provide additional context to the queries. While this may not be as expressive as retrieval augmentation, this provides a good compromise of greatly reducing sequence length while giving up only a little in performance. We do so by using a small subset of the unlabeled data. As shown empirically, a LLM teacher trained in this manner performs significantly better than a non-RA LLM trained directly on the human data, while at the same time allowing us to efficient distillation.

In the second stage we use the *Teacher* LLM to annotate the entire unlabeled dataset. This is in turn used to train the final *Student* model that will be used in practice.

5 Experiments and Results

Experimental Setup: Our experiments were all conducted using the mT5 (Xue et al., 2021) checkpoints. We validate performance across three learning rates (1e-3, 1e-4, 5e-5) – selecting the best checkpoint using the validation set loss. For mod-

els trained from the provided training sets, we used a batch size of 64 in our experiments and trained for 4K steps (EComm) / 20K steps (Orcas-I).

For distilled models, we used a batch size of 128 for Teacher models and 1024 for Student models. We use different batch sizes because of the model architectures, mT5 for the Teacher vs a BERT-based model for the Student. In both cases, we trained for 1 epoch, unless mentioned otherwise. We only use the encoder of the mT5 model with an additional layer added on top to predict the classification scores. The Professor, the Teacher and the Student fine-tuning experiments are all set up as a query intent classification task. Given that the Teacher and Student models are trained on millions of examples and this itself is a time and resource intensive step, we restrict our experiments to only one epoch. We demonstrate performance gains even with one epoch via the techniques elaborated in this paper.

We studied the effect of distillation data size, for both stages of distillation. For the EComm dataset, we used an in-house retriever to find related documents. For Orcas-I, we use the provided docids (aggregated at per-query level) for retrieval augmentation. Unless specified, we use (upto) the top-10 results for retrieval augmentation¹. Sequence length for models are based on the training set and features (set to 99%-percentile of sequence lengths).

Students and Features: Our experiments demonstrate results for a fast, efficient 4-layer transformer student architecture, with hidden dimensionality of 256. We default to using the query as the only feature in the student for simplicity. To compare against query expansion techniques, we used a sophisticated in-house memorization-based query expansion model in our Professor / Teacher experiments on EComm. This expansion model – which we refer to as **ExpandTerms** – provides a list of related terms for a given query, which are concatenated with the query (and identifiers for start / end of each feature).

Metrics: To compare performance of different models we use two metrics: **MicroF1** and **MacroF1** for Orcas-I, and **AUC-PR** and **AUC-ROC** for EComm. For EComm, we only report

¹For Orcas-I, nearly 2/3rd of the queries only have a single provided result, while some have upwards of 2000 results, which is why the lengths for RA features on Orcas-I in Table 2 are smaller. The 10 results augmented are randomly chosen if more exist.

Model	Size	ROC	PR
query	Base	0.0%	0.0%
+ RA (titles, urls)	Base	+4.3%	+4.6%
query	XL	+2.7%	+3.1%
+ RA (titles, urls)	XL	+6.3%	+6.7%
query	XXL	+3.0%	+3.3%
+ RA (titles, urls)	XXL	+6.4%	+6.9%

Table 3: Results demonstrating the benefit of Retrieval Augmentation (RA) across all model sizes.

EComm	ROC	PR
query	0.0%	0.0%
+ Terms	+2.6%	+1.9%
+ RA (titles)	+4.8%	+4.8%
+ RA (titles) + Terms	+5.1%	+5.2%
+ RA (urls)	+5.3%	+5.7%

Table 4: Analysis of the impact of different features (using Base-sized models) for the EComm dataset. ExpandTerms abbreviated as Terms.

performance of models relative to the mT5 query-only Base-sized model².

5.1 Effect of Retrieval Augmentation

While the use of retrieval augmentation (RA) has been known to improve query classification performance (Broder et al., 2007), the benefit of RA is unclear in the age of LLMs. Thus, we start by evaluating the first stage of QUILL *i.e.*, the *RA model*. As seen in Table 3, RA improves performance significantly across all model sizes including the billion-parameter+ XL and XXL models. In fact the gains from RA on the Base-sized model exceed the gains obtained by increasing model size of a query-only model to XXL. Given the gains observed across all models sizes, we use Base-sized models in the rest of the paper to simplify experimentation.

²For a sense of scale, each 0.5% point increase in metrics on EComm is considered a significant gain.

EComm	ROC	PR
Orcas-I	MicF1	MacF1
query	69.8	69.75
+ RA (titles)	+6.3%	+5.1%
+ RA (urls)	+8.2%	+6.2%
+ RA (titles+urls)	+9.0%	+7.2%

Table 5: Analysis of the impact of different features (using Base-sized models) for the Orcas-I dataset.

EComm	ROC	PR
Baseline Teacher (Finetuned on Training Set)	+2.6%	+1.9%
QUILL Teacher (2M Prof Distilled Set)	+3.3%	+2.8%
QUILL Teacher (4M)	+3.4%	+2.9%
QUILL Teacher (8M)	+3.5%	+2.9%
QUILL Professor	+5.3%	+5.7%

Table 6: Comparison of different Teacher models trained directly or via Professor-distillation for the EComm dataset.

A natural question that may arise though is how do these gains from RA compare to those obtained by powerful query expansion techniques. Thus, we performed an in-depth ablation of features for the EComm dataset (on a Base-sized model for ease of experimentation) as seen in Table 4 and for the Orcas-I dataset as seen in Table 5. These results clearly demonstrate the potency of powerful query expansion models (*i.e.*, **ExpandTerms**) – as evidenced by the large $\sim 2\%$ gains over query-only models. However, we find that RA adds even more value over these highly sophisticated expansion models with an a nearly 5+% increase in performance. Furthermore, we find that RA techniques can still be combined with query expansion for further gains.

The improvements on RA for Orcas-I (seen in Table 5) are even more substantial, with a nearly 9% improvement over the query-only baseline, via the use of the titles and urls of related documents. Interestingly, among RA features we find that urls tend to perform slightly better than titles on both datasets. We believe this to be because titles can have a higher variance of informativeness – with both highly verbose and very short titles commonly seen. Hence, given the simplicity and consistency of urls, we chose to use RA(urls) for subsequent experiments as the *Professor* model.

5.2 Distilling gains from RA

We next focus on the second stage of QUILL: Distilling the RA model. Typically larger amounts of distillation data lead to better performance. However, given the increased sequence length of RA models and the cost of retrieval augmentation itself, annotating large distillation sets is highly challenging. Thus to capture such practical trade-offs, we

Orcas-I	MicF1	MacF1
Baseline Teacher (Finetuned on Training Set)	69.8	69.75
QUILL Teacher (2M Prof Distilled Set)	+1.1%	+0.8%

Table 7: Comparison of different Teacher models trained directly or via Professor-distillation for the Orcas-I dataset.

only used a small subset of the unlabeled data for the *QUILL Professor to Teacher* distillation. In particular, we used 4M examples for EComm (*i.e.*, 3.1% of unlabeled data) and 2M for Orcas-I (19%) for this first stage of distillation – to represent a set that is small enough set to be practical, but large enough to learn from. However, we do share results for varying this size to understand its importance.

QUILL *Teacher* models were thus trained by distilling the RA(urls) *Professor* models. Our *Teacher* models had the same capacity and architecture (*i.e.*, *mT5-Base* as the *Professor*³ – except it does not use RA (features).

As a realistic and competitive baseline, we chose a *Baseline Teacher* that resembles the *QUILL Teacher* in all aspects bar one – the data they are trained on. Specifically, the *Baseline Teacher* is directly trained from the gold-labeled training data, unlike the *QUILL teacher*. We believe this is representative of practical applications today, where LLMs are trained directly on gold-labeled sets (before being distilled into the final student models). To further challenge QUILL, we leverage the powerful **ExpandTerms** features (for the EComm dataset) in our *Teacher* models – both *Baseline* and *QUILL*. We believe this provides a more challenging but realistic evaluation setup, since many baseline models in use today avail of powerful features (along with the query).

As seen from the results in Table 6 and Table 7, we find the *QUILL Teachers* provide a significant performance improvement over the *Baseline Teacher*, despite having never directly seen the gold label data. On EComm, despite using an enhanced (realistic) baseline, *QUILL teachers* are $\sim 1\%$ better on all metrics. We find a similar gap on Orcas-I despite the *Teacher* there being trained on only 2M examples (just 1.5x the training set size). Put differently, we now have trained our non-retrieval aug-

³We observed similar trends even if the *Teacher* had less capacity than the *Professor*.

Model (# Distillation)	ROC	PR
No Distillation Student	-6.3%	-7.3%
Baseline Student	-0.9%	-1.6%
QUILL Student	+2.0%	+1.5%
QUILL 1-Stage Student(4M)	+0.4%	-0.2%
QUILL 1-Stage Student(32M)	+1.1%	+0.6%

Table 8: Performance of the different student models trained from different teachers and using differing amounts of distillation data (on EComm).

mented language model to benefit from the gains of retrieval augmentation. Even though the student model does not have Retrieval Augmentation, because of the Teacher model’s performance improvement, it is possible to annotate a considerably large number of training examples. We observe the Student models to close the gap (compared to the Teacher) given larger training datasets.

To test the robustness of QUILL teachers we also varied the amount of distillation data used – halving or doubling it. While there still exist distillation gaps to the professor (which can be narrowed via more distillation data) on both datasets, our proposed approach works well even when using small amounts of distillation data – which in turn allows us to save significant compute.

Query	Example URL	W/L
bengals	sports.yahoo.com/nfl/teams/cin/	✓
pah compounds	en.wikipedia.org/wiki/Polycyclic_aromatic_hydrocarbon	✓
launch tech usa	launchtechusa.com/	✓
noun university	en.wikipedia.org/wiki/Noun	✗
airbed uk	www.airbnb.co.uk/	✗

Table 9: Wins/losses examples on Orcas-I.

5.3 Final student training

So far, we have shown that QUILL can learn a better (non-RA) teacher. However, an important question remains unanswered: Can these Teacher gains be translated to the final student model? In particular, we postulate that the predictions of the QUILL Teacher may be more robust and easier to learn (for student models) than those of the Baseline. To verify this hypothesis we compared 4 fast student

models (4-layer encoder-only models), with the only difference being the data they were trained on:

- *No Distillation Student*: This is the simple solution of directly training the Student using the labeled data.
- *Baseline Student*: This is the current standard involving distilling the Baseline Teacher model using the full unlabeled set.
- *QUILL Student*: This is the proposed solution involving distilling the QUILL Teacher model using the full unlabeled set.
- *QUILL 1-Stage Student*: Rather than the two stage distillation approach, this student is directly distilled from the Professor using a subset of the unlabeled data.

As seen from Table 8, all QUILL-based students significantly outperform the Baseline Student. In particular our proposed 2-stage approach leads to a ~ 3 point gain on both metrics. This is notable in that the gap between QUILL and Baseline students is even higher than the Teachers – which we attribute to the QUILL Teacher labels being more robust.

Comparing different QUILL students, we find that there is a notable performance gain by first distilling into a non-RA teacher, before distilling into the final student. While 1-stage distillation performance improves as more data is used, even when 1/4th of all unlabeled data is retrieval-augmented and annotated by the Professor for direct distillation, it still falls short of the 2-stage approach. Together, these results show: (1) QUILL students outperform the current state-of-the-art significantly, and (2) QUILL benefits from the 2 stage distillation of Professor to Teacher to final student.

5.4 Examples of Wins/Losses from RA

While the previous sections focused on demonstrating the efficacy (and efficiency) of QUILL, we wanted to also understand **why** and **where** are some of these gains from RA stem from. To do so we used the test-set of Orcas-I and sampled illustrative examples of wins / losses (Table 9) between the baseline and the retrieval-augmented professor models. One common win pattern we found for RA models is when the query is unclear, or uses technical terms / abbreviations. In these cases, the augmented urls / titles help provide additional context for the language model to understand what the

query is about. On the flip side, we also found the biggest loss pattern to be when retrieval was inaccurate, which in turn misled the model regarding the query intent. For example, we found our retriever returned wikipedia results more often than it should, which misled the model to believe the query had *Factual* intent.

6 Future Work

While we studied the problem of query intent classification in this paper, the approach proposed in our paper is general and could be applied to any query understanding task. Following our approach, could enable myriad query understanding tasks use retrieval augmentation in a practically realistic and efficient manner. We leave this to future work though. We should also note that our experiments reveal non-trivial distillation gaps in both stages of distillation, which we believe is another open opportunity for future improvements.

7 Conclusion

This paper provides a practical recipe for combining Retrieval Augmentation and Large Language Models. In particular, we proposed QUILL as an approach to tackle the problem of query intent classification. Our empirical study demonstrates conclusively that Retrieval Augmentation can provide significant value over existing approaches. Furthermore we show that via our two-stage distillation approach, that QUILL not only learns better performing, more robust teachers, but also leads to even bigger gains when distilled into fast, real-world capable production student models.

8 Acknowledgements

We sincerely thank Jiecao Chen, William Dennis Kunz, Austin Tarango, Lee Gardner, Yang Zhang, Constance Wang, Derya Ozkan, Nitin Nalin, Raphael Hoffmann, Iftekhhar Naim, Siddhartha Brahma, Siamak Shakeri, Hongkun Yu, John Nham, Ming-Wei Chang, Marc Najork, Corinna Cortes and many others for their insightful feedback and help. We also thank the EMNLP Reviewers for their thorough review, feedback and suggestions.

Limitations

This paper focuses on efficient and effective way of improving query intent classification using Retrieval Augmentation (RA) and Multi-stage distillation. While we have made the best attempts to ensure a robust and efficient method, we would be remiss to not point out some key limitations of our work:

- **Quality of Retrieval:** A key reason for the gains seen in this paper is the use of Retrieval Augmentation. This additional context provided in the form of result titles / URLs are helpful, but are dependent on the quality of the retrieval system. While we did not get a chance to explore the dependence of performance gains on retrieval quality, we plan to explore this in future work.
- **Dependency of Retrieval:** While our approach provides for a practical and low-compute way of incorporating retrieval augmentation, it still does add some compute (to augment the datasets) and system complexity. While we considered this trade-off well worth it in our use case, this may depend on specific settings.
- **Retrieval-Augmentation techniques:** As discussed in Section 4, we used a simple concatenation based retrieval augmentation. However, there do exist more sophisticated techniques for retrieval augmentation. For example, models built on a Fusion-in-Decoder (Izacard and Grave, 2020) backbone have demonstrated great performance (Hofstätter et al., 2022b; Izacard et al., 2022) and improved efficiency (Hofstätter et al., 2022a). We believe that these more sophisticated retrieval-augmentation technique may bring further improvements in our system and leave this for future work to follow up on.
- **Datasets:** The lack of large public query sets means that we were very limited in terms of what public benchmarks we could study this problem on. While ORCAS-I is the largest such available set, they lack many alternatives that are large enough to study the effects of distillation. In the future though, we hope to use the (somewhat related) problem of question-answering where larger datasets (with large

enough unlabeled data) exist for a more thorough study.

- **Distillation gaps:** Our results also clearly demonstrate large distillation gaps in both stages. While there have been innovative techniques proposed to improve distillation performance, we intentionally chose to keep things simple as those approaches are largely complementary to the problem we study in this work.
- **Limited "Large" Model Experiments:** While our work is intended for and positioned in the context of "Large" Language Models, we realize that our most common model choice (mT5-Base), may not be the most representative model in that category. This was an intentional choice on our end as we hoped doing so would make the work more relevant to use cases and applications with more limited compute. For practitioners interested in models with tens of billions of parameters, we refer them to our analysis of mT5-XXL sized models in Table 3, that demonstrates the viability of our approach on models of that scale.

Ethics Statement

In this paper, we used only publicly available Language Model and Checkpoints that have been previously published – namely mT5.

An important consideration when working with query datasets is data privacy. This is perhaps the biggest reason why there do not exist many large public query datasets. We intentionally chose ORCAS-I for this reason, as it is constructed from the ORCAS query set – which is widely regarded as a well-constructed, non-PII, sufficiently anonymized query dataset. While the EComm dataset used in this paper is proprietary, we should note that it too has been scrubbed of PII and aims to follow the same (if not higher) data privacy principles. Our data (and methodology) do not contain any information for or target any demographic or identity characteristics.

The task we focus on – query classification – is a general problem that benefits everyone. In fact, it can enable better IR systems thereby benefiting users who otherwise might not get answers. Thus, we do not anticipate any biases or misuse issues stemming from this. We believe that by using

publicly available and vetted retrieval models, the resulting retrieval augmented models should not create any new or further any existing biases.

In many ways a goal of our work is making retrieval augmentation more practical and reducing compute needs for any such applications. While we did present results with XXL sized models, we focused most of our experiments on the smaller, more efficient Base-sized models so as to benefit a wider section of our community and to reduce the computational needs of our experiments.

References

- Daria Alexander, Wojciech Kusa, and Arjen P. de Vries. 2022. **ORCAS-i**. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.
- Ricardo Baeza-Yates, Liliana Calderón-Benavides, and Cristina González-Caro. 2006. The intention behind web queries. In *International symposium on string processing and information retrieval*, pages 98–109. Springer.
- Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pages 2206–2240. PMLR.
- Andrei Broder. 2002. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM New York, NY, USA.
- Andrei Z. Broder, Peter Ciccolo, Marcus Fontoura, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. 2008. **Search advertising using web relevance feedback**. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM '08*, page 1013–1022, New York, NY, USA. Association for Computing Machinery.
- Andrei Z Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 231–238.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.
- Nick Craswell, Daniel Campos, Bhaskar Mitra, Emine Yilmaz, and Bodo Billerbeck. 2020. **Orcas: 20 million clicked query-document pairs for analyzing search**. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management, CIKM '20*, page 2983–2989, New York, NY, USA. Association for Computing Machinery.
- Fernando Diaz and Donald Metzler. 2006. Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161.
- Alejandro Figueroa. 2015. Exploring effective features for recognizing the user intent behind web queries. *Computers in Industry*, 68:162–169.
- Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Papatat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.
- Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. 2020. **Learning-to-rank with bert in tf-ranking**. *arXiv preprint arXiv:2004.08476*.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).
- Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2022a. Fid-light: Efficient and effective retrieval-augmented text generation. *arXiv preprint arXiv:2209.14290*.
- Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2022b. Multi-task retrieval-augmented text generation with relevance sampling. *arXiv preprint arXiv:2207.03030*.
- Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.
- Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

- Bernard J Jansen, Danielle L Booth, and Amanda Spink. 2008. Determining the informational, navigational, and transactional intent of web queries. *Information Processing & Management*, 44(3):1251–1266.
- In-Ho Kang and GilChang Kim. 2003. Query type classification for web document retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71.
- Ashish Kathuria, Bernard J Jansen, Carolyn Hafernik, and Amanda Spink. 2010. Classifying the user intent of web queries using k-means clustering. *Internet Research*.
- Dirk Lewandowski, Jessica Drechsler, and Sonja Von Mach. 2012. Deriving query intents from web search engine queries. *Journal of the American Society for Information Science and Technology*, 63(9):1773–1788.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198.
- Alaa Mohasseb, Mohamed Bader-El-Den, and Mihaela Cocea. 2019. A customised grammar framework for query classification. *Expert Systems with Applications*, 135:164–180.
- Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docttttquery. *Online preprint*, 6.
- Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.
- 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*.
- Xuanhui Wang. 2020. *Query Segmentation and Tagging*, pages 43–67. Springer International Publishing, Cham.
- Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. *mT5: A massively multilingual pre-trained text-to-text transformer*. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.
- Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng. 2020. *Query understanding via intent description generation*. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM '20*, page 1823–1832, New York, NY, USA. Association for Computing Machinery.

Distinguish Sense from Nonsense: Out-of-Scope Detection for Virtual Assistants

Cheng Qian[†], Haode Qi[†], Gengyu Wang[†], Ladislav Kunc[†], Saloni Potdar[‡]

[†]IBM Watson, [‡]Apple Inc.

{cheng.qian, haode.qi, gengyu, lada}@ibm.com, s_potdar@apple.com

Abstract

Out of Scope (OOS) detection in Conversational AI solutions enables a chatbot to handle a conversation gracefully when it is unable to make sense of the end-user query. Accurately tagging a query as out-of-domain is particularly hard in scenarios when the chatbot is not equipped to handle a topic which has semantic overlap with an existing topic it is trained on. We propose a simple yet effective OOS detection method that outperforms standard OOS detection methods in a real-world deployment of virtual assistants. We discuss the various design and deployment considerations for a cloud platform solution to train virtual assistants and deploy them at scale. Additionally, we propose a collection of datasets that replicates real-world scenarios and show comprehensive results in various settings using both offline and online evaluation metrics.

1 Introduction

In the context of task-oriented dialog, Out of Scope (OOS) detection is the problem of identifying end-user queries that are beyond the scope of a chatbot. While this problem is generally studied under the umbrella of “out of domain” detection in machine learning, we show that unique challenges arise in real-world applications. We study this problem in the context of our enterprise virtual assistant (VA) platform which is used by 10,000+ customers to design chatbots. In this setting, the natural language understanding models comprising of In Scope (IS) and OOS detection modules, need to determine whether an input query belongs to a set of pre-defined intents or if it is out of scope for the chatbot.

Real-world success of OOS systems often involves measuring how good they are at *containment*, i.e., the user queries are resolved and contained by the chatbot while minimizing human interventions. Since containment rate can be only observed after launching the VA online, offline

metrics such as IS accuracy and OOS accuracy are needed while designing and developing the models.

The average designer using an enterprise VA platform is not a machine learning expert. This leads to a variety of challenges in the provided user data, which constitutes the need for robust algorithms. Firstly, end-users often provide data that is heavily imbalanced or noisy for both IS and OOS detection.

While designing VA for enterprise use-cases, IS and OOS examples often naturally belong to the same domain. Such OOS samples are called In Domain OOS (ID-OOS) as opposed to Out-of-Domain OOS (OOD-OOS) which are relatively easier OOS samples from a different domain (Zhang et al. (2022)). Designers expect the VA to detect these relevant but unsupported topics (ID-OOS) even though it has high semantic overlap with IS examples. Finally, while entities defined by the designer play an important role for a real-world VA, they are often ignored in academic OOS settings. We show that entities must be modeled in conjunction with IS and OOS classification.

In this paper, we discuss the challenges of designing a real-world OOS detection system in depth and common approaches taken to design such a system. We propose a simple but effective algorithmic modification for OOS detection in a real-world deployed system. This system models entities, intent and OOS classification jointly and addresses the challenges around data. We propose a comprehensive benchmark based on public datasets and show that our method outperforms standard approaches while being simple to deploy and maintain.

2 Challenges

2.1 Metrics

Containment and Disambiguation For businesses, the key performance index (KPI) metric is typically different from the common machine learning metrics used to test the performance of

the algorithm. Businesses use containment rate to measure chatbot performance - the portion of conversations not handed off or escalated to a human agent for quality reasons. Among offline evaluation metrics, IS performance provides the best estimate for containment rate. Disambiguation is a mechanism to increase containment by asking end-users clarification questions and providing more than one relevant intent. This has to be counter-balanced with high OOS performance so that we don't provide a set of predictions in the form of IS classes for an OOS query. This is essential to appear "intelligent" and handle conversations gracefully.

In Domain Out of Scope Detection refers to detecting OOS samples with high semantic overlap with IS examples in the same domain (ID-OOS). ID-OOS queries are often harder to detect than the easier Out of Domain OOS (OD-OOS) samples. The algorithm should be able to identify ID-OOS and also generalize well to unseen OOD-OOS.

2.2 Data Considerations

The average designer of an enterprise VA platform doesn't need to have ML background, hence the expectations of labeled data are very different from an ML expert end-user.

Class Imbalance is often extreme in data provided by VA designer, with some intent classes having more number of examples than others.

Data-scarce scenarios Labeling data is often expensive for enterprises who desire good performance with very few labelled examples per class, and often no OOS labeled data.

Noisy data Unlike public datasets, enterprise datasets have semantically similar intents due to overlap in business use cases. Additionally, real-world end-user input queries to VAs usually contain spelling errors, intentionally repeated characters, emojis, and slang. Proper normalization is required to improve robustness of OOS algorithms.

2.3 Computational Efficiency

While developing the OOS detection algorithm for an enterprise VA platform, we need to strike a good trade-off between cost of serving the model and performance of the model. Based on our experience, VA platforms are expected to handle training sets of more than 10k training examples and more than 1000 classes.

Model size & memory: There are over 100,000 customer-specific models deployed in production and each chatbot serves millions of queries per

month. Hence low maintenance, training and inference costs can increase profitability.

Training time: Designers typically make changes in an iterative fashion, designing the VA through trial and error. For an interactive experience in the product, the OOS detection component needs to train in 1 minute (Qi et al. (2020)).

Inference time: During the inference, each query passes through all the natural language understanding (NLU) components - IS classification, OOS detection, entity recognition and spellchecking, and needs to provide the predictions in 10 milliseconds.

2.4 Entities and OOS Detection

Entities are designed to represent nouns from end-user inputs and are crucial for VAs to respond accordingly and haven't been studied extensively in OOS detection.

Terminologies Designers can define entities with special terminologies that are out of vocabulary of any other public or private corpus. This requires OOS detection methods to differentiate such terminologies from gibberish sentences.

Synonyms The OOS detection algorithm is expected to produce similar detection scores across the multitudes of synonyms of the same entity.

Numeric Values System entities like date, number, time etc. are pre-configured in a VA to cover a wide range of concepts. However, there is no one-size-fits-all solutions for system entities. E.g., the system entity "11" can be a part of domain specific terminology "operating system Windows 11". The OOS detection algorithm needs to be aware of these system entity values and decide the relevance of the sentence based on the context.

We introduce several potential solutions for handling entities in OOS detection and analyze their advantages and disadvantages.

Concatenation of all entity synonyms In the context of Binary OOS detection, we add one synthetic IS example in to the training data by concatenating all entity synonyms provided for a chatbot. Context independent features such as uni-grams, bi-grams and mean/max pooling of word-embeddings will help recognizing these entities as IS at the runtime. This simple approach works well empirically but has the disadvantage of ignoring the context and semantic meaning.

Synonyms and Entity proxies in intent templates In enterprise VA, an entity can be defined with multiple synonyms. In our product, we support entity proxies, which is a definition of a certain entity and its associated synonyms that are considered

equal. This greatly simplifies training data definition at the cost of potential instability at runtime: our intent detection and OOS algorithm should return the same confidence and same predicted label if one synonym is replaced by another. For the example in Table 3, if "cell phone" is defined as an entity proxy, VA designer only references the symbol "cell phone" in training examples, and at runtime "i want an iphone 11" gets the exact same prediction as "i want an iphone xr".

3 OOS Detection Algorithms

OOS detection algorithms can be broadly classified into single-stage and multi-stage.

3.1 Single-stage OOS

All the IS classes and optionally the OOS class are used together train a single model to determine if the incoming query belongs to one of the IS intents or is OOS.

Multiclass Classification In this approach, the algorithm treats the OOS examples as an additional class as explored in (Zhan et al. (2021), Choi et al. (2021)), alongside the IS classes to train a multi-class classification model for both IS intent detection and OOS detection.¹ This approach trains a single algorithm for both OOS detection and IS detection tasks. In practice, this approach is susceptible to over-fitting to the provided OOS examples and might not generalize well to unseen OOS queries. Additionally, it can fail in the presence of severe class-imbalance.

In-scope Classification utilizing output distribution This type of methods trains a classifier on IS data which outputs a probability vector with low maximum probability or high entropy for an OOS input, as explored in Lewis and Gale (1994), Hendrycks and Gimpel (2016), Lee et al. (2018a), Yilmaz and Toraman (2022). These methods train a single model for IS detection and applies a threshold on output probability distribution statistics (such as max and entropy) for OOS detection. However, in practice, training data typically has semantically overlapped intents which will mislead the system and increase unnecessary human agent intervention as shown in Table 1.

3.2 Multi-stage OOS

Multi-stage OOS method uses a binary classifier to determine if a query is IS or OOS in the first stage.

¹<https://docs.microsoft.com/en-us/azure/cognitive-services/luis/concepts/intents#none-intent>

In the subsequent stages we determine which of the IS intent is the closest match.

Binary Classification (IS/OOS): A binary classifier is trained using the IS examples and OOS examples as explored in Tax and Duin (1999). The classification result is used to determine if end-user query is OOS or ID. In case there are no OOS training examples, the binary OOS classifier can be replaced with an one-class classifier or other unsupervised methods. Another solution for the lack of OOS training data is synthetic OOS training examples, refer to Section 4 for more discussion.

In-scope Classification plus unsupervised methods on internal (hidden state) representation trains a classifier based on IS training examples, and utilizes internal representation (for example, concatenation of hidden states from several layers of a neural network) of the IS classifier for an unsupervised OOS detection algorithm, like auto-encoder with reconstruction loss, distance based approach (Wu et al., 2022), (Shen et al., 2021), and density based approach (Lin and Xu, 2019).

3.3 Our Approach

We show a simple modification to the multi-stage OOS to improve the performance of the system and alleviate problems with the other approaches mentioned previously. Our approach **Binary Classification(In Scope/OOS) discounting on In Scope scores** treats OOS classification as a binary classification problem like the previous formulation. However, the binary classification score of the OOS detection algorithm is used to discount the IS classification score to determine the final IS score (more details to follow). In case no training OOS examples are available our OOS detection algorithm becomes one-class classification. This formulation is related to calibration (Kamath et al., 2020) that trains a new model to reject inputs when the model is over-confident. However, our approach applies the OOS output as discounting factor instead of binary score leading to better performance in the context of enterprise VA as shown in Table 4.

In terms of the OOS classification component, we implement a distance based approach based on sentence embedding of both IS and OOS training examples (if labeled). At training time, we first apply the trick described in Section 2.4 to preprocess entities among other text normalization steps, then query the sentence embeddings from a sentence encoder. For each IS example, we store the linear combination of its sentence embedding and the mean embedding of its corresponding intent

class in an approximate nearest neighbor(ANN) search index. If there are OOS training examples, we store their sentence embeddings in the same ANN search index. At runtime, a query is pre-processed the same way as training examples, the cosine distance to the nearest neighbor will be used as OOS score to discount the output from the IS classifier. If the nearest neighbor is OOS, we add an additional constant to the corresponding nearest distance, to discount the confidence more. The discounting step uses the OOS score cos_dist and IS classifier output confidence **conf**, we apply the formula below to produce the final output confidence vector **final_conf** as follows:

$$\mathbf{final_conf} = (1 - f(\max(cos_dist, 0))) \cdot \mathbf{conf} \quad (1)$$

$$f(x) = \begin{cases} x & x \geq 0.5 \\ sigmoid(a \cdot (x - 0.5)) & \text{otherwise} \end{cases} \quad (2)$$

, where a is a constant that can be tuned for different applications. The motivation for Formula 2 is to reduce the amount of discounting on IS confidence (comparing to a linear discounting function), when OOS classifier predicts a low cosine distance (thus high similarity) for an utterance.

Typically, a fixed threshold T on the final output confidences is used in real world applications to determine whether an input utterance is predicted as IS or OOS: a new input is deemed OOS if its final output confidence is less than T . Theoretically, this threshold is not critical to machine learning metrics, especially threshold independent metrics. Even for threshold dependent metrics, this threshold can always be tuned in accordance with the scale of final output confidence to achieve the same results. However in practice, as a commercial VA platform, a fixed threshold reduces the maintenance cost of a chatbot and only a small fraction of the chatbot designers will try to tune the threshold. In our product, 0.2 threshold is set as the default value.

3.4 Benchmark Dataset

We collect 8 intent classification datasets to comprehensively evaluate the methods mentioned above regarding the challenges, including IS classification, OOS detection, and scalability. The 8 datasets include **ATIS** (Hemphill et al., 1990), **BANKING77** (Casanueva et al., 2020), **CLINC150** (Larson et al., 2019), **StackOverflow**², **SNIPS** (Coucke et al., 2018), **HAR** (Liu et al., 2019), **ROSTD** (Gangal et al., 2020), and **HINT3** (Arora et al., 2020). A

²https://storage.googleapis.com/download.tensorflow.org/data/stack_overflow_16k.tar.gz

Query	Intent
I need assistance with my retirement account	retirement account
I need to talk to a agent about my retirement account	agent

Table 1: The two queries shown are semantically overlapped. For the query "I need to talk to a assistant about my retirement account", the correct intent should be "agent" but one can expect "retirement account" and "agent" having similar probability. For approaches that rely on probability vector to detect OOS input, these examples can mislead them to treat valid IS queries as OOS.

summary of dataset statistics after preprocessing is provided in Table 2.

To evaluate methods' performance on ID-OOS detection, we ensure all datasets contain ID-OOS examples. For datasets that only contain IS examples, we randomly choose a number of IS intents and treat them as OOS so that the number of examples in these intents are about 25% of the whole training dataset. The full list of chosen intents for each dataset are listed in Appendix A.1.

We reorganize some of the datasets as follows. CLINC150 includes 2 domains, banking and credit card, we evaluate them separately along with the full set. For StackOverflow, 10% examples in training set is stratified splitted as dev set. For HAR, we remove examples with missing 'answer', and stratified split remaining examples into train, dev, and test set with a 80, 10, and 10 percentage. Different from other selected datasets, ROSTD contains 4,000 OOS examples. ROSTD-coarse is the version that only keep higher hierarchical intent types. Examples in "reminder" intent type from original ROSTD-coarse are treated as ID-OOS. HINT3 consists of 3 domains, including SOFMatress, Curekart and Powerplay11, so we evaluate them separately. 10% of training examples in each of HINT3 datasets is stratified split as dev set.

3.5 Evaluation metrics

Based on current literature, there are 2 types of commonly used metrics for OOS detection.

Threshold dependent metrics are metrics calculated with predicted labels e.g. accuracy. These metrics compare the probability score against a threshold to determine whether a query is considered OOS or not. Also threshold dependent metrics encourage joint evaluation of intent detection and OOS detection that are more suitable under the

Dataset	Train			Dev			Test		
	IS	ID-OOS	OOD-OOS	IS	ID-OOS	OOD-OOS	IS	ID-OOS	OOD-OOS
CLINC150-FULL	11300	3700	100	2260	740	100	3390	1110	1000
CLINC150-BANKING	400	100	0	400	100	600	400	100	1350
CLINC150-CREDIT	400	100	0	400	100	600	400	100	1350
ATIS	4053	425	0	458	42	0	812	81	0
BANKING77	6533	2089	0	1160	380	0	2320	760	0
Stack Overflow	5400	1800	0	600	200	0	6000	2000	0
SNIPS	9371	3713	0	500	200	0	500	200	0
ROSTD	23621	6900	0	3238	943	1500	6661	1960	3090
ROSTD-coarse	23621	6900	0	3238	943	1500	6661	1960	3090
HAR	15893	4592	0	1986	575	0	1985	576	0
HINT3 (SOFMattress)	229	66	0	26	7	0	158	73	166
HINT3 (Powerplay11)	317	102	0	38	14	0	244	31	708
HINT3 (Curekart)	415	125	0	45	15	0	390	62	539

Table 2: **Dataset Statistics.** We preprocess all datasets (details in A.1) and numbers reflect their sizes.

context of VA. Following the literature (Wu et al. (2022), Zhou et al. (2022), Zeng et al. (2021)), the threshold dependent metrics are listed here:

Overall Accuracy is the percentage of examples being correctly classified. For an IS query, it’s predicted correctly if and only if the predicted IS label is correct and the query is predicted as IS. For an OOS query, it should be predicted as OOS to make a correct prediction. **IS Accuracy** is the percentage of correctly predicted IS examples out of all IS examples. **IS F1** is the macro averaged F1 scores across all IS intents. **OOS F1** is the F1 score for OOS examples.

Threshold independent metrics are metrics calculated with a vector of scores each of which measures how confident or likely an OOS algorithm considers a query irrelevant. Such a score is often a number between 0 and 1 where 1 represents IS and 0 represents OOS. This paper follows the literature (Shen et al. (2021), Ryu et al. (2018), Liang et al. (2017), Lee et al. (2018b)) and defines IS as the positive class and OOS as the negative class. We use the metrics for evaluating OOS detection performance: **FPRN**, where N is an integer between 0 and 100, is the false positive rate(FPR) when the true positive rate(TPR) is at least N%. A false positive is an OOS example predicted as IND. We use FPR90 and FPR95. **AUROC** is the area under the Receive Operating Characteristic curve, which measures TPR against FPR at different thresholds. **AUPR_IN** and **AUPR_OUT** are metrics measuring area under the precision-recall curve, when IS and OOS are considered as the positive class, respectively.

<i>Training Example</i>
Can I buy a cell phone ?
<i>Training Entities</i>
entity: cell phone
synonyms: iphone, samsung, galaxy, iphone XR, iphone 11, etc..
<i>Inference Queries</i>
A galaxy is a huge collection of gas, dust, stars and their solar systems.
what is the latest model of galaxy s series?

Table 3: VA designer defines the entity "cell phone" with synonyms. The 1st query contains the word "galaxy" but it is OOD-OOS. The 2nd with "galaxy" is ID-OOS.

3.6 Experiments and Results

We conduct experiment on the benchmark datasets to compare different OOS problem formulations listed in Section 3 (Our discounting approach, Binary Classification (IS/OOS), Multiclass Classification³, and IS Classification utilizing output distribution). As a comparison for the OOS problem formulation only, we keep the IS classification algorithm and OOS algorithm same as our production setup across the 4 formulations, without focusing on the exact choice or implementation of the IS and OOS algorithms. For the discounting method, we use our production intent detection and OOS detection as is. For multi-class classification method, we consider OOS examples as an additional IS intent. For the IS Classification utilizing output distribution formulation, we train an IS classifier with IS training examples and take the max of output confidence vector as the OOS score.

Offline Evaluation Table 4 reports the simple average of metrics across all our benchmark datasets.

³Threshold independent metrics for Multiclass classification is omitted, as our IS classifier outputs confidence vectors (which do not sum up to 1) instead of predicted probability, thus it involves no such component as OOS scores.

Method	Overall Acc.	IS Acc.	IS F1	OOS F1	OOS recall	FPR90	FPR95	AUROC	AUPR_IN	AUPR_OUT
Binary	82.45	86.92	77.87	76.84	74.18	22.76	30.29	91.70	91.21	88.16
Multiclass	74.34	90.36	72.81	74.08	64.84	-	-	-	-	-
IS clf + Max	79.17	85.93	77.99	70.02	65.15	32.66	44.45	87.53	87.15	80.27
Discounting (Our Approach)	84.70	86.43	79.71	80.63	79.31	20.07	27.20	92.64	91.07	89.90

Table 4: **Performance on all datasets** This table compares the discounting method against binary classification, multiclass classification, the IS classifier + max confidence on the full test sets.

Method	Overall Acc.	IS Acc.	IS F1	OOS F1	OOS recall	FPR90	FPR95	AUROC	AUPR_IN	AUPR_OUT
Binary	53.45	72.70	47.97	45.26	38.40	59.24	78.80	81.37	72.07	86.98
Multiclass	52.97	74.65	49.44	56.15	41.15	-	-	-	-	-
IS clf + Max	54.36	75.13	53.46	57.90	43.66	56.38	73.74	76.97	60.33	86.01
Discounting (Our Approach)	61.46	69.38	51.79	60.05	54.94	50.84	71.02	82.83	69.32	88.97

Table 5: **Performance on HINT3** This table compares the discounting method against the Multiclass classification method, the binary classification method, the IS classifier + max confidence on the full test sets.

The discounting approach achieves better performance across most metrics. We report the average metrics in Table 5 on the subset of the 3 HINT3 datasets, which are designed to represent real world imbalanced datasets.

Table 6 compares the Multi-Class formulation against our formulation on all datasets. Our approach performs on par on ID-OOS but generalize better to OOD-OOS. In real world application, limited ID-OOS is provided by customer during training but the algorithm is expected to perform well on both categories without overfitting.

Method	Test set	Overall Acc.	IS Acc.	IS F1	OOS F1	OOS recall
K+1 Classes	IND+ID-OOS	90.65	90.36	86.46	90.82	92.76
discounting	IND+ID-OOS	86.14	88.04	84.62	76.25	80.81
K+1 Classes	IND+OOD-OOS	60.28	89.41	65.07	46.60	32.69
discounting	IND+OOD-OOS	78.31	85.72	76.71	74.36	71.43
K+1 Classes	IND+both OOS types	74.34	90.36	72.81	74.08	64.84
discounting	IND+both OOS types	84.70	86.43	79.71	80.63	79.31

Table 6: **Performance on various test sets** We compare the discounting method against the Multiclass classification method on 3 versions of test sets: IS + ID-OOS, IS + OOD-OOS (average across the datasets with OOD-OOS test examples), IS + both types of OOS examples.

Online Evaluation During real-world deployment of this algorithm, we conducted additional online evaluation by analyzing the output distribution change on real production traffic because chatbot designers typically rely on output confidence scores to make business decisions eg. jumping to a node in the dialog tree, handing off to human agents or asking a follow-up question. Therefore, we deployed the proposed OOS algorithm in production and monitored different statistics on 10% of randomly selected real traffic for months before surfacing it to end-users. We observed that more than 85% of live traffic queries will have a less than 10% change in top confidence after the change in OOS algorithms

(Full distribution shown in Figure 1 in Appendix). For enterprise customers with complex dialog conditions, a new algorithm that does not disrupt the normal workflow is critical for adoption.

Computational Efficiency and Scalability Our product has a training set size limit of 25k IS and OOS training examples each and 2k IS classes. Based on this maximum training set size setting, the maximum model size for OOS detection is less 150MB based on offline testing. Based on online testing, the 99 percentiles for training time and model size of our OOS algorithm are within 30 seconds and 70MB, respectively.

4 Conclusion

The paper presents a novel Out of Scope (OOS) detection component in a task-oriented dialog system. It allows the assistant to recognize user input that is not designed to be answered by the assistant and need to be handed off to a human agent. For business, a well designed Out of Scope detection system can improve customer satisfaction, user engagement, lead generation and saves cost. On one hand, business wants the assistant to hand off quickly when a user input is Out of Scope. On the other hand, unnecessary hand off could increase human intervention and reduce the value of VA. We design an OOS detection system that overcomes a multitude of real-world challenges, and deploy it in production.⁴ We list out the lessons learned and both offline and online evaluation techniques for designing a robust, efficient and scalable system for enterprise VA platform.

⁴<https://cloud.ibm.com/docs/assistant?topic=assistant-irrelevance-detection>, <https://cloud.ibm.com/docs/assistant?topic=assistant-release-notes#assistant-jun162022>

Limitations

Extensive benchmarking for other languages is out-of-scope for this work, but we have extended the approach to many European languages in the product with similar gains in performance (Wang et al., 2022). Code switching isn't evaluated in this work, but it is commonly observed in chatbots deployed in the wild.

We have not discussed synthetic OOS examples. Despite its demonstrated effectiveness, they need caution in real world production system from a robustness perspective: it's possible to introduce spurious correlation by generated synthetic data.

References

- Gaurav Arora, Chirag Jain, Manas Chaturvedi, and Krupal Modi. 2020. Hint3: Raising the bar for intent detection in the wild. *arXiv preprint arXiv:2009.13833*.
- Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. [Efficient intent detection with dual sentence encoders](#). In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at <https://github.com/PolyAI-LDN/task-specific-datasets>.
- DongHyun Choi, Myeongcheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. [Outflip: Generating out-of-domain samples for unknown intent detection with natural language attack](#). *CoRR*, abs/2105.05601.
- Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. [Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces](#). *CoRR*, abs/1805.10190.
- Varun Gangal, Abhinav Arora, Arash Einolghozati, and Sonal Gupta. 2020. Likelihood ratios and generative classifiers for unsupervised out-of-domain detection in task oriented dialog. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7764–7771.
- Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.
- Dan Hendrycks and Kevin Gimpel. 2016. [A baseline for detecting misclassified and out-of-distribution examples in neural networks](#). *CoRR*, abs/1610.02136.
- Amita Kamath, Robin Jia, and Percy Liang. 2020. [Selective question answering under domain shift](#). *CoRR*, abs/2006.09462.
- Stefan Larson, Anish Mahendran, Joseph J Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K Kummerfeld, Kevin Leach, Michael A Laurenzano, Lingjia Tang, et al. 2019. An evaluation dataset for intent classification and out-of-scope prediction. *arXiv preprint arXiv:1909.02027*.
- Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018a. [Training confidence-calibrated classifiers for detecting out-of-distribution samples](#). In *International Conference on Learning Representations*.
- Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018b. [A simple unified framework for detecting out-of-distribution samples and adversarial attacks](#). In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.
- David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR '94*, pages 3–12, London. Springer London.
- Shiyu Liang, Yixuan Li, and R. Srikant. 2017. [Principled detection of out-of-distribution examples in neural networks](#). *CoRR*, abs/1706.02690.
- Ting-En Lin and Hua Xu. 2019. [Deep unknown intent detection with margin loss](#). *CoRR*, abs/1906.00434.
- Xingkun Liu, Arash Eshghi, Pawel Swietojanski, and Verena Rieser. 2019. Benchmarking natural language understanding services for building conversational agents. *arXiv preprint arXiv:1903.05566*.
- Haode Qi, Lin Pan, Atin Sood, Abhishek Shah, Ladislav Kunc, and Saloni Potdar. 2020. [Benchmarking intent detection for task-oriented dialog systems](#). *CoRR*, abs/2012.03929.
- Seonghan Ryu, Sangjun Koo, Hwanjo Yu, and Gary Geunbae Lee. 2018. [Out-of-domain detection based on generative adversarial network](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 714–718, Brussels, Belgium. Association for Computational Linguistics.
- Yilin Shen, Yen-Chang Hsu, Avik Ray, and Hongxia Jin. 2021. [Enhancing the generalization for intent classification and out-of-domain detection in SLU](#). *CoRR*, abs/2106.14464.
- David M. J. Tax and Robert P. W. Duin. 1999. [Data domain description using support vectors](#). In *ESANN 1999, 7th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 21-23, 1999, Proceedings*, pages 251–256.
- Gengyu Wang, Cheng Qian, Lin Pan, Haode Qi, Ladislav Kunc, and Saloni Potdar. 2022. Benchmarking language-agnostic intent classification for virtual assistant platforms. In *Proceedings of the Workshop on Multilingual Information Access (MIA)*, pages 69–76.

- Yanan Wu, Keqing He, Yuanmeng Yan, QiXiang Gao, Zhiyuan Zeng, Fujia Zheng, Lulu Zhao, Huixing Jiang, Wei Wu, and Weiran Xu. 2022. [Revisit overconfidence for OOD detection: Reassigned contrastive learning with adaptive class-dependent threshold](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4165–4179, Seattle, United States. Association for Computational Linguistics.
- Eyup Halit Yilmaz and Cagri Toraman. 2022. [D2U: distance-to-uniform learning for out-of-scope detection](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 2093–2108. Association for Computational Linguistics.
- Zhiyuan Zeng, Keqing He, Yuanmeng Yan, Zijun Liu, Yanan Wu, Hong Xu, Huixing Jiang, and Weiran Xu. 2021. [Modeling discriminative representations for out-of-domain detection with supervised contrastive learning](#). *CoRR*, abs/2105.14289.
- Li-Ming Zhan, Haowen Liang, Bo Liu, Lu Fan, Xiaoming Wu, and Albert Y. S. Lam. 2021. [Out-of-scope intent detection with self-supervision and discriminative training](#). *CoRR*, abs/2106.08616.
- Jianguo Zhang, Kazuma Hashimoto, Yao Wan, Zhiwei Liu, Ye Liu, Caiming Xiong, and Philip Yu. 2022. [Are pre-trained transformers robust in intent classification? a missing ingredient in evaluation of out-of-scope intent detection](#). In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 12–20, Dublin, Ireland. Association for Computational Linguistics.
- Yunhua Zhou, Peiju Liu, and Xipeng Qiu. 2022. [KNN-contrastive learning for out-of-domain intent classification](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5129–5141, Dublin, Ireland. Association for Computational Linguistics.

A Appendix

A.1 List of IN-OOS intents

Here we list the intents for each dataset that are treated as IN-OOS intents in our benchmark.

Stackoverflow: python

SNIPS: SearchCreativeWork and SearchScreeningEvent

HAR: intents, hue_lightoff, explain, remove, addcontact, wemo_on, podcasts, createoradd, music, praise, radio, dontcare

ROSTD: reminder/set_reminder, reminder/cancel_reminder, reminder/show_reminders

HINT3 SOFMattress:
SIZE_CUSTOMIZATION,
ABOUT_SOF_MATTRESS, LEAD_GEN,
COMPARISON, WARRANTY, DELAY_IN_DELIVERY

HINT3 Powerplay11:
NO_EMAIL_CONFIRMATION,
TEAM_DEADLINE, FAKE_TEAMS,
CANNOT_SEE_JOINED_CONTESTS, REFUND_OF_ADDED_CASH, HOW_TO_PLAY,
FEEDBACK, ACCOUNT_NOT_VERIFIED,
DEDUCTED_AMOUNT_NOT_RECEIVED,
CRITICISM, NEW_TEAM_PATTERN, OFFERS_AND_REFERRALS

HINT3 Curekart: EXPIRY_DATE,
CONSULT_START, CHECK_PINCODE,
ORDER_TAKING, INTERNATIONAL_SHIPPING, IMMUNITY,
SIDE_EFFECT, START_OVER, PORTAL_ISSUE, MODES_OF_PAYMENTS, ORDER_QUERY, SIGN_UP, WORK_FROM_HOME

A.2 Our OOS problem formulation is algorithm-agnostic:

We conducted the same experiment with another OOS algorithm: autoencoder with reconstruction loss as OOS score. The findings are similar: our OOS formulation demonstrate advantages over others. Detailed metrics are shown in Table 7.

A.3 Online evaluation statistics

Figure 1 shows the full distribution of differences in top confidence between the proposed OOS algorithms vs previous OOS algorithm on a percentage of live traffic

Distribution of difference in top confidences between 2 OOS algorithm:

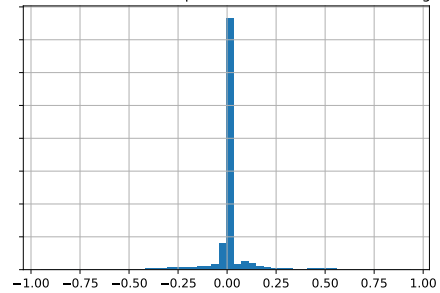


Figure 1: Distribution of differences in top confidence between the proposed OOS algorithms vs previous OOS algorithm on a percentage of live traffic

Method	Overall Acc.	IS Acc.	IS F1	OOS F1	OOS recall	FPR90	FPR95	AUROC	AUPR_IN	AUPR_OUT
Binary	83.25	85.09	76.68	81.70	77.16	28.24	34.78	91.47	90.37	89.27
Multiclass	74.34	90.36	72.81	74.08	64.84					
IS clf + Max	79.17	85.93	77.99	70.02	65.15	32.66	44.45	87.53	87.15	80.27
Discounting (Our Approach)	84.30	85.85	77.88	83.17	79.35	19.65	25.56	93.18	92.24	91.63

Table 7: **Performance metrics** This table compares the discounting method against the Multiclass classification method, the binary classification method, the IS classifier + max confidence on the full test sets, using autoencoder as the OOS detection algorithm.

PLATO-Ad: A Unified Advertisement Text Generation Framework with Multi-Task Prompt Learning

Zeyang Lei*, Chao Zhang*, Xinchao Xu, Wenquan Wu, Zheng-Yu Niu, Hua Wu, Haifeng Wang, Yi Yang and Shuanglong Li

Baidu Inc., Beijing, China

{leizeyang, zhangchao38, xuxinchao, wuwenquan01}@baidu.com

{niuzhengyu, wu_hua, wanghaifeng, yangyi15, lishuanglong}@baidu.com

Abstract

Online advertisement text generation aims at generating attractive and persuasive text ads to appeal to users clicking ads or purchasing products. While pretraining-based models have achieved remarkable success in generating high-quality text ads, some challenges remain, such as ad generation in low-resource scenarios and training efficiency for multiple ad tasks. In this paper, we propose a novel unified text ad generation framework with multi-task prompt learning, called PLATO-Ad, to tackle these problems. Specifically, we design a three-phase transfer learning mechanism to tackle the low-resource ad generation problem. Furthermore, we present a novel multi-task prompt learning mechanism to efficiently utilize a single lightweight model to solve multiple ad generation tasks without loss of performance compared to training a separate model for each task. Finally, we conduct offline and online evaluations. Experiment results show that PLATO-Ad significantly outperforms the state-of-the-art on both offline and online metrics. PLATO-Ad has been deployed in a leading advertising platform with 3.5% CTR improvement on search ad descriptions and 10.4% CTR improvement on feed ad titles.

1 Introduction

In recent years, online advertising has been regarded as one of the most popular ways of internet monetization. A captivating and persuasive ad can greatly improve the probability of users clicking the ads or purchasing recommended products (Jansen and Resnick, 2005). Thus, advertisers usually spare no effort to improve the quality of displayed ads. Traditionally, some advertisers may manually design ads for high quality. However, this approach suffers from low efficiency and high labor costs. Some other works design pre-defined templates (Fujita et al., 2010; Thomaidou et al., 2013)

*Equal contribution.



Figure 1: Example ads in search ad and feed ad systems.

or use recurrent neural networks (RNN) (Hughes et al., 2019) to automatically create ads. However, these ads generated by templates are generally rigid and not unappealing enough to users. Over 15% ads generated by RNN are reported to be non-sense or bad (Hughes et al., 2019), which hinders the application of this method in real-world business scenarios with high-quality standards.

Recently, with the remarkable success of the pretraining plus fine-tuning paradigm, some works utilize pretraining-based natural language generation methods to generate content-rich, diverse and attractive ads based on large-scale corpora (Wang et al., 2021; Zhang et al., 2021a,b). Although these methods achieve significant success in generating high-quality ads, they still face great challenges when applied in real-world commercial scenarios. **Firstly**, the performance of these pretraining-based models relies heavily on large-scale high-quality training corpora, which may be yet difficult to be obtained in some low-resource¹ scenarios. **Secondly**, these work usually train a separate model for each task respectively when applied in multiple ad tasks, e.g., ad title and description generation (Wang et al., 2021; Zhang et al., 2021a), copywriting generation (Zhang et al., 2021b), selling point generation (Guo et al., 2021). This is inefficient and expensive for multiple ad tasks that require multiple copies of the model’s parameters, with also ignoring the associations between multiple ad generation tasks (Lester et al., 2021).

¹Here, low resource denotes that relatively less training data can be available.

To this end, we propose a novel unified text ad generation framework with multi-task prompt learning, called PLATO-Ad, to tackle the aforementioned problems. Concretely, the model architecture of PLATO-Ad is a Transformer-based pre-trained language model with 12 transformer blocks. **To effectively address the low-resource ad generation problem**, we propose a three-phase transfer learning mechanism to train PLATO-Ad. Specifically, we first pre-train PLATO-Ad on generic-domain text corpus to equip the model with the ability to generate fluent natural sentences. Then, in the second phase, we consecutively post-pretrain PLATO-Ad on the datasets of multiple resource-rich ad generation tasks where massive data is available and open-domain question answering (QA) datasets, which enables the model to learn to generate ad-domain and commonsense-enriched text. Finally, we use the prompting mechanism (Liu et al., 2021) to transfer the well-trained PLATO-Ad in the second phase to low-resource ad generation tasks. In this way, we can generate high-quality ads for those low-resource ad generation scenarios that lack a large amount of high-quality human-written data. Furthermore, **to improve training efficiency and reduce application costs for multiple ad tasks**, in the post-pretraining phase, we propose a novel multi-task prompt learning mechanism by introducing task prompts and multiple losses to better fuse multiple tasks into one single model without loss of performance.

The main contributions are summarized as follows:

- We present a unified ad generation framework with multi-task prompt learning, named PLATO-Ad, to effectively tackle the ad generation in low-resource scenarios and training efficiency for multiple resource-rich ad tasks. To our knowledge, this work is the first to study low-resource ad generation with a prompting mechanism in industrial scenarios.
- We propose a novel three-phase transfer learning mechanism to address ad generation in low-resource settings by transferring from generic domain text generation to resource-rich ad domain text generation, and finally to low-resource ad text generation.
- Furthermore, we devise a novel multi-task prompt learning mechanism by introducing task prompts and multiple training objectives

to efficiently fuse multiple resource-rich ad tasks into a single lightweight model without loss of performance.

- The offline and online experiment results show PLATO-Ad significantly outperforms the state-of-the-art models and can generate high-quality ads in low-resource settings. In the A/B test, the advertisements generated by PLATO-Ad would bring about 3.5% CTR improvement on search ad descriptions and 10.4% CTR improvement on feed ad titles.

2 Related Work

Previous works on text ad generation rely on designing pre-defined templates to construct readable ad sentences (Bartz et al., 2008; Fujita et al., 2010; Thomaidou et al., 2013). However, these template-based ads are generally rigid and not diverse enough leading to unappealing to users. Afterward, Hughes et al. (2019) presents data-driven methods to learn to write text ads from existing examples. The model employs LSTMs and attention layers to encode product landing pages and decode text ads. Moreover, REINFORCE with baseline (Rennie et al., 2017) is leveraged to generate attractive ads that potentially have a larger click rate. This method achieves a certain degree of success in the automatic generation of text ads. However, over 15% text ads it generates are labeled as non-sense, broken, or bad, which fails to meet the high-quality standard for production (Hughes et al., 2019).

Recently, some work utilize pretraining-based natural language generation methods to generate content-rich, diverse and attractive ads based on large-scale corpus (Wang et al., 2021; Zhang et al., 2021a,b; Wei et al., 2022). However, these methods require large-scale high-quality training corpus and low training efficiency when applied to multiple ad generation scenarios. In this paper, we propose a unified text ad generation framework with multi-task prompt learning to tackle ad generation problems in low-resource settings and improve training efficiency for multiple resource-rich ad tasks.

3 Methodology

3.1 Problem Settings

Different ad generation tasks, such as ad description generation, ad title generation, selling point generation and tips generation (Li et al., 2019) for

Task	Input Text	Auxiliary Attributes	Output Text	Low-Resource	Training Phase
Dialog Gen.	Context	-	Response	✗	Pretrain
Ad Desc. Gen.	Ad Title	Product Landing Page (Text)	Ad Desc.	✗	Post-Pretrain
Ad Title Gen.	Product Entity	Product Landing Page	Ad Title	✗	Post-Pretrain
Sel. Point Gen.	Ad Title	Product Attributes	Product Selling Points	✗	Post-Pretrain
Comment Gen.	Product Desc.	Sentiment Polarity	Comment	✗	Post-Pretrain
QA	Question	Keyword	Answer	✗	Post-Pretrain
Commonsense-enriched Ad Desc. Gen.	Ad Title	Product Landing Page & Keyword	Commonsense-rich Ad Desc.	✓	Prompting
Tips Gen.	Ad Title	Focus Point & Sentiment Polarity	Tips	✓	Prompting

Table 1: Description of Different Ad Generation Tasks. Here, Gen. denotes generation.

recommendation, usually contain different inputs and outputs. To simplify description, we generalize elements of all ad generation tasks into the following fields: input text, auxiliary attributes and output text. Table 1 shows inputs and outputs of different ad generation tasks.

Formally, we refer to input text as $X = (x_1, x_2, \dots, x_n)$, auxiliary attributes $A^j = (a_1^j, a_2^j, \dots, a_k^j)$, $j = 1, 2, \dots, l$ and output text as $Y = (y_1, y_2, \dots, y_m)$. Here, A^j represents a word sequence of the j -th ad attribute, n, k, m denotes the sequence length of input text, the j -th ad attribute and output text respectively. l denotes the number of auxiliary attributes. $x_i, a_i^j, y_i \in \mathcal{V}$ denotes a word token. \mathcal{V} denotes the vocabulary. The ad generation process can be defined as a sequence-to-sequence generation task as follow:

$$\begin{aligned}
 Y &\sim p(Y|X, A) \\
 &= p(y_1, y_2, \dots, y_m | x_1, x_2, \dots, x_n; A^1, A^2, \dots, A^l).
 \end{aligned}
 \tag{1}$$

The goal of PLATO-Ad is to learn this function $p(Y|X, A)$.

3.2 Model Framework

3.2.1 Overview

PLATO-Ad is a transformer-based pretraining model with a three-phase transfer learning mechanism, as shown in Fig 2. Concretely, we first pre-train PLATO-Ad on a large-scale open-domain dialog corpus following the setting of PLATO-2 (Bao et al., 2021), a Chinese dialog-oriented pretraining model. The purpose of this step is to enable PLATO-Ad to generate fluent natural sentences and model correlation from input (context) to output(response). Then, in the second phase, we continue to pretrain PLATO-Ad on the datasets of multiple resource-rich ad generation tasks with relatively massive training data available and open-domain QA datasets, which makes PLATO-Ad learn generating ad-domain and commonsense-enriched texts. This step enables that PLATO-Ad

can transfer learning from generic domain text generation to ad domain text generation. Moreover, to apply PLATO-Ad to multiple ad generation scenarios more efficiently, we design a multi-task prompt learning mechanism by introducing task prompts and multiple losses to efficiently fuse multiple tasks into a single lightweight model without loss of performance. Finally, in the third phase, we use task prompts to transfer well-trained PLATO-Ad to low-resource text ad generation tasks. To simplify the description, we call the first phase as **Generic-Domain Pretraining**, the second phase as **Ad-Domain Post-Pretraining**, and the third phase as **Low-Resource Prompting**. We elaborate the details of all phases in the following.

3.2.2 Generic-Domain Pretraining

We first pretrain PLATO-Ad on generic-domain text following the settings of PLATO-2 (Bao et al., 2021). Specifically, the pretraining dataset is collected from public social medias, which contains 1.2B (context, response) samples. Unlike the original PLATO-2 with multi-step losses, we train PLATO-Ad only with negative log-likelihood (NLL) loss for fluent natural text generation.

3.2.3 Ad-Domain Post-Pretraining

Then, we post-pretrain PLATO-Ad on datasets of multiple resource-rich ad generation tasks and high-quality open-domain QA datasets to equip PLATO-Ad with the ability to generate ad-domain and commonsense-enriched text. In particular, the post-pretraining datasets contain four real-world resource-rich ad generation tasks, such as ad description generation, ad title generation, selling point generation and comment generation. The construction and data preprocessing of these datasets can be found in Section A.1.

Meanwhile, to more efficiently train lightweight PLATO-Ad for fusing multiple ad tasks, we design a multi-task prompt learning mechanism by introducing task prompts and multiple training ob-

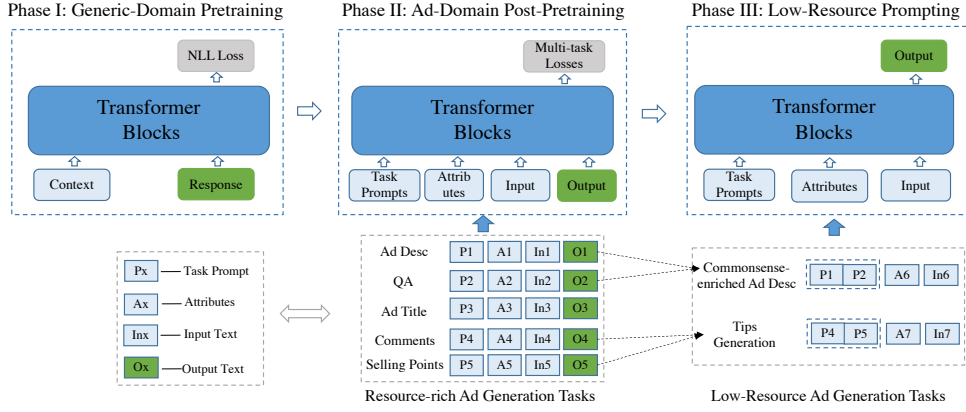


Figure 2: The architecture of our PLATO-Ad framework.

jectives. The model infrastructure is shown in Fig 2, which consists of input representation, transformer blocks and multi-task training objectives.

Input Representation. For each token of input text, auxiliary attributes and output text, its representation is the sum of the token embedding, the type embedding and the position embedding. Type embeddings are employed to differentiate inputs from different parts and position embeddings are set according to the token position in each sentence. Two special tokens [BOS] and [EOS] are inserted at the beginning and end of sentences for separation. Meanwhile, to effectively integrate multiple different tasks into one model, we add task prompts as the part of inputs following prompt tuning (Lester et al., 2021). Here, different tasks correspond to different task prompts and we randomly initialize the representation vectors of task prompts.

Multiple Training Objectives. To better train PLATO-Ad for meeting different ad tasks, we design multiple training losses to optimize the repetition problem, relevance of input and output, and controllability in these tasks.

a) Unlikelihood Loss for Repetition. To alleviate the repetition problem in text generation, especially long text generation for ad desc task, we adopt an unlikelihood loss following (Welleck et al., 2020). Formally, the unlikelihood loss L_U can be defined as follow.

$$L_U = - \sum_{y_t \in C^t} \log(1 - p_\theta(y_t | y < t, X, A)) \quad (2)$$

Where y_t denotes t -th output token, C^t refers to a set of negative candidate tokens (i.e. repetitive tokens), X and A denotes input text and attributes.

b) Relevance Loss. To improve relevance between input and output, we design a discrimina-

tive relevance loss L_R to estimate the relevance between them, which can be defined as follow.

$$L_R = - \log p(l = 1 | X, Y^+) - \log p(l = -1 | X, Y^-) \quad (3)$$

where $p(l = 1)$ denote positive samples and $p(l = -1)$ denote negative samples. Positive samples are from golden $\langle X, Y \rangle$ pairs and negative samples are obtained by random sampling $\langle X, Y \rangle$ pairs.

c) Keyword Loss for Controllability. Some ad generation tasks, for example, selling point generation and tip generation, expect the generated response to contain auxiliary attributes. Therefore, we design keyword loss L_C following (Kumar et al., 2021) to improve the token-level controllability of PLATO-Ad.

$$L_C = \min_{i=1}^m (- \log p(y_i = K | X, A)) \quad (4)$$

where K denotes the keyword controlled to generate. Finally, we train PLATO-Ad by summing multiple losses.

$$L = L_N + \lambda_u L_U + \lambda_r L_R + \lambda_c L_C \quad (5)$$

where L_N denotes negative log-likelihood (NLL) loss, $\lambda_u, \lambda_r, \lambda_c$ are model hyperparameters.

3.2.4 Low-Resource Prompting

After post-pretraining, we use prompting methods to apply well-trained PLATO-Ad to low-resource ad generation scenarios. Specifically, we select two low-resource ad generation, commonsense-enriched ad description generation (Zhang et al., 2021a) and tips generation (Li et al., 2019), both of which lack large-scale high-quality training

data. Then we achieve task-level transfer by using the combination of task prompts from resource-rich ad tasks. In particular, the task prompt of commonsense-enriched ad description generation is a combination of task prompts of ad description generation and open-domain QA tasks, as shown in Fig 2. In this way, we will transfer common sense from open-domain QA to ad descriptions to generate commonsense-rich ad descriptions. Similarly, the task prompt of tips generation is a combination of task prompts of selling point generation and comment generation, which results in generating informative reviews for product selling points.

4 Experiments

4.1 Datasets

To our knowledge, there are no publicly available large-scale high-quality ad-domain datasets and we collect post-pretraining datasets from a leading advertising platform. Table 2 shows the statistics of these datasets. More details about dataset construction and data preprocessing can be found in Appendix A.1.

	Train	Dev	Test
Ad Desc. Gen.	10,800,000	50,000	10,000
Ad Title Gen.	4,288,167	50,000	10,000
Sel. Point Gen.	798,686	50,000	10,000
Comment Gen.	5,777,279	50,000	10,000
QA	17,859,294	50,000	10,000

Table 2: Dataset statistics.

4.2 Baselines

We select the following baselines to evaluate the effectiveness of our model.

CHASE (Zhang et al., 2021a): It is the online state-of-the-art model deployed on the commercial ad systems. We follow the same model settings.

PLATO-2-FT: It directly uses the PLATO-2 (Bao et al., 2021) model to finetune on multiple ad datasets. We use the released parameters ².

PLATO-Ad: It is our proposed model in this paper with a multi-task prompting mechanism and three-phase transfer learning. These three models have the same magnitude of parameters (about 90M), which ensures a fair comparison.

²<https://github.com/PaddlePaddle/Knover/tree/luge-dialogue>

4.3 Evaluation Metrics

We use Perplexity (PPL) (Brown et al., 1992) and Pairwise-BLEU (Shen et al., 2019) to automatically measure the model quality and diversity of generation results. The more diverse the hypothesis set is, the lower the Pairwise-BLEU is. Meanwhile, we conduct a manual evaluation on 200 random samples from our test dataset. Three participants were recruited to measure the quality of the result generated by each baseline from three perspectives, including Readability (Read.), Relevance (Rele.), Information (Info.). Each perspective is measured by a 3-point Likert question where 0 is bad, 1 is neutral and 2 is good. The Overall (Over.) score is the average value of the above three scores. The detailed evaluation metrics can be found in appendix A.4.

4.4 Experimental Results on Resource-rich Ad Generation Tasks

We conduct a set of experiments to evaluate the effectiveness of PLATO-Ad on resource-rich ad generation tasks. As shown in Table 3, PLATO-Ad significantly outperforms the state-of-the-art CHASE and PLATO-2-FT in terms of all the metrics. It demonstrates that PLATO-Ad can generate more fluent (lower PPL) and more diverse (lower Pairwise-BLEU) ads in comparison with baselines. Meanwhile, PLATO-Ad removes multi-task losses causes more high PPL and Pairwise-BLEU scores, **indicating the effectiveness of multi-task losses.**

In addition, we conduct experiments to verify the efficiency of the multi-task prompt learning mechanism. From Table 4, we find that single lightweight PLATO-Ad obtains better performance than multiple separate models trained on different tasks. This demonstrates **the effectiveness of fusing multiple tasks into a single model with shared parameters via multi-task prompt learning mechanism.**

Models	PPL↓	Pairwise-BLEU↓
CHASE	8.30	47.90
PLATO-2-FT	3.29	41.93
PLATO-Ad	2.21	38.46
PLATO-Ad w/o multi losses	2.36	38.67

Table 3: Results on the test set of ad description generation. PLATO-Ad w/o multi losses denote that PLATO-Ad removes multi-task losses. Bold scores are the best.

Datasets	PLATO-Ad w/o Task Prompt	PLATO-Ad
Ad Desc Gen.	2.50	2.21
Ad Title Gen.	2.81	2.65
Sel. Point Gen.	7.31	6.70
Comment Gen.	6.40	6.42
QA	3.71	3.31

Table 4: PPL on test sets of multiple ad generation tasks. PLATO-Ad w/o Task Prompt represents that PLATO-Ad removes the multi-task prompting mechanism and separately trains on each corresponding ad generation task.

4.5 Transfer Learning on Low-Resource Ad Generation Tasks

We investigate the effectiveness of PLATO-Ad on two low-resource³ ad generation tasks, commonsense-enriched ad description generation and tips generation. We compare the manual evaluation results of PLATO-2-FT (separately finetuning on these two datasets) and PLATO-Ad Prompting on 200 random samples of our test datasets. From Table 5, we can see that PLATO-Ad surpasses PLATO-2-FT on all manual metrics (especially Info.) for these two low-resource ad generation tasks, indicating that PLATO-Ad can generate more relevant and commonsense-enriched ad descriptions and tips. This also verifies **the effectiveness of three-phase transfer learning**.

Figure 3 shows the examples generated by PLATO-2-FT and PLATO-Ad. We can see that PLATO-Ad generates more commonsense-rich ad descriptions and more informative tips. *We analyze that this is because PLATO-Ad realizes the transfer learning of multiple tasks in the post-pretraining phase, such as transferring ad description and QA tasks to the commonsense-enrich ad description generation task and transferring comment generation and selling point tasks to the tip generation task.* Meanwhile, we also find that PLATO-Ad can generate less repetitive and more controllable ads in comparison with PLATO-2-FT.

4.6 Online A/B Test

We also conduct online A/B tests in a leading advertising platform to show the superiority of PLATO-Ad. We used 5% real-world web traffics in the advertising platform to conduct the A/B test. We had already gotten the permission from advertiser

³We manually evaluated only about 9.3% of ad descriptions dataset (93/1000) are commonsense-enriched ad descriptions and 8.6% of comment generation dataset (86/1000) are suitable for tips generation.

Tasks	Models	Read.	Rele.	Info.	Over.
Com.A.	PLATO-2-FT	1.70	1.25	0.68	1.21
	PLATO-Ad	1.91	1.89	1.98	1.93
Tip.	PLATO-2-FT	1.96	1.68	1.62	1.75
	PLATO-Ad	1.96	1.86	1.93	1.92

Table 5: Manual evaluation results on Commonsense-enriched Ad Description Generation (Com.A.) and Tip Generation (Tip.) tasks. The Kappa value is above 0.4, showing modest agreement among evaluators.

Commonsense-enriched Ad Desc	Input: 孩子注意力不集中怎么办? (What if children do not pay attention?) Attributes: 雨林教育从事3-14岁少年注意力培养和逻辑训练(Rainforest Education Corporation focuses on the attention and logic training of children aged 3-14.)
PLATO-2-FT	雨林教育,专注3-14岁少儿的注意力培养,逻辑思维训练,帮助孩子提升注意力,逻辑思维能力训练。(Rainforest Education Corporation, focuses on the attention training and logical thinking ability training of children aged 3-14, and helps children improve their attention, and logical thinking ability training.)
PLATO-Ad	孩子注意力不集中,首先要培养孩子良好的生活习惯,让孩子养成早睡早起,多参加一些户外活动,培养孩子的专注力。雨林教育专业培训3-14岁孩子注意力和逻辑,值得信赖。(If the child is inattentive, first of all, it is necessary to cultivate the child's good living habits, let the child develop to go to bed early and get up early, participate in some outdoor activities, and cultivate the child's concentration. Rainforest Education is a trustworthy company that professionally trains the attention and logic for children aged 3-14.)
Tip Gen.	Input: 男士纯棉休闲裤,限时打折优惠(Men's cotton slacks with discount for a limited time) Attributes: Focus Points (价格#质地 Price # Material); Sentiment Polarity (正面Positive)
PLATO-2-FT	挺不错的,应该买的人挺多的。(It's pretty good, and there should be a lot of people buying it.)
PLATO-Ad	这款男士休闲裤,纯棉的,穿着很舒服,同时质量也不错,男士穿上帅气又减龄,价格也便宜。(This pure cotton men's casual pant is very comfortable to wear. Meanwhile, the quality is also good, and men wear it to look handsome and age-reducing, and the price is also very favorable.)

Figure 3: Example ads generated by PLATO-Ad and PLATO-2-FT. PLATO-2-FT could generate repetitive ads (green color). Compare with PLATO-2-FT, PLATO-Ad can generate commonsense-rich ads (red color) and more controllable ads (blue color).

to use PLATO-Ad to generate advertisement descriptions/titles for this test. This online A/B test lasts for one week. On each day there were about 1 million page views (with ad shows) for the testing. We use Click-Through Rate (CTR) (Richardson et al., 2007) and Conversion Rate (CVR) (Lee et al., 2012) compared with CHASE to show the improvement of PLATO-Ad. Except for the displayed ad descriptions/titles, we keep other settings the same. Table 6 demonstrates that PLATO-Ad can bring more significant CTR and CVR improvement compared with the state-of-the-art CHASE. The details about deployed workflow can be found in Appendix A.5.

Tasks	Δ CTR	Δ CVR
Search Ad Description	+3.5%	+1.7%
Feed Ad Title	+10.4%	+4.1%
Feed Selling Point	+7.6%	+2.3%

Table 6: Online A/B Testings on different ad generation tasks. $\frac{\Delta\text{CTR/CVR}}{\text{CTR/CVR of CHASE}} = \frac{\text{CTR/CVR of PLATO-Ad} - \text{CTR/CVR of CHASE}}{\text{CTR/CVR of CHASE}}$

4.7 Conclusion

In this paper, we propose a novel unified text ad generation framework with multi-task prompt learning, called PLATO-Ad, to tackle universal

commercial ad generation tasks. Experiments show that PLATO-Ad can generate commonsense-rich and relevant ads in low-resource scenarios via a **three-phase transfer learning mechanism** and improve training efficiency for multiple resource-rich ad tasks by using a **multi-task prompt learning mechanism** to fuse multiple tasks into a single lightweight model without loss of performance. In the future, we will extend the idea of PLATO-Ad to more real-world text generation tasks.

Ethics Statement

We make sure that we have the copyright to use all datasets to train and deploy. Meanwhile, these datasets do not contain any user’s private information. In manual evaluation, we ensure that all annotators were treated fairly. This includes but is not limited to, compensating them fairly, ensuring that they were able to give informed consent, and ensuring that they were voluntary participants who were aware of any risks of harm associated with their participation. During A/B testing and system deployment, all generated advertisements must be approved by the advertiser before using. Before online deployment, we conduct a post-processing procedure for all generated advertisements, including the basic correlation filtering (quality control) and business risk control system to strictly control the exposure risk of the displayed advertisements. Meanwhile, for badcases or harmful contents that are found or fed back from customers when displayed online, we also have an online blacklist procedure to filter them in real time.

References

- Siqi Bao, Huang He, Fan Wang, Hua Wu, Haifeng Wang, Wenquan Wu, Zhen Guo, Zhibin Liu, and Xinchao Xu. 2021. Plato-2: Towards building an open-domain chatbot via curriculum learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2513–2525.
- Kevin Bartz, Cory Barr, and Adil Aijaz. 2008. Natural language generation for sponsored-search advertisements. In *Proceedings of the 9th ACM Conference on Electronic Commerce*, pages 1–9.
- Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, Jennifer C Lai, and Robert L Mercer. 1992. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Atsushi Fujita, Katsuhiko Ikushima, Satoshi Sato, Ryo Kamite, Ko Ishiyama, and Osamu Tamachi. 2010. Automatic generation of listing ads by reusing promotional texts. In *Proceedings of the 12th International Conference on Electronic Commerce: Roadmap for the Future of Electronic Business*, pages 179–188.
- Xiaojie Guo, Shugen Wang, Hanqing Zhao, Shiliang Diao, Jiajia Chen, Zhuoye Ding, Zhen He, Yun Xiao, Bo Long, Han Yu, et al. 2021. Intelligent online selling point extraction for e-commerce recommendation. *arXiv preprint arXiv:2112.10613*.
- J Weston Hughes, Keng-hao Chang, and Ruofei Zhang. 2019. Generating better search engine text advertisements with deep reinforcement learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2269–2277.
- Bernard J Jansen and Marc Resnick. 2005. Examining searcher perceptions of and interactions with sponsored results. In *Workshop on Sponsored Search Auctions*.
- Shachi H. Kumar, Hsuan Su, Ramesh Manuvinakurike, Saurav Sahay, and Lama Nachman. 2021. [Controllable response generation for assistive use-cases](#). *CoRR*, abs/2112.02246.
- Kuang-chih Lee, Burkay Orten, Ali Dasdan, and Wentong Li. 2012. Estimating conversion rate in display advertising from past performance data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 768–776.
- Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.
- Piji Li, Zihao Wang, Lidong Bing, and Wai Lam. 2019. Persona-aware tips generation? In *The World Wide Web Conference*, pages 1006–1016.
- Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.
- Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7008–7024.
- Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Tianxiao Shen, Myle Ott, Michael Auli, and Marc’Aurelio Ranzato. 2019. Mixture models for diverse machine translation: Tricks of the trade. In *International conference on machine learning*, pages 5719–5728. PMLR.

Stamatina Thomaidou, Ismini Lourentzou, Panagiotis Katsivelis-Perakis, and Michalis Vazirgiannis. 2013. Automated snippet generation for online advertising. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1841–1844.

Xiting Wang, Xinwei Gu, Jie Cao, Zihua Zhao, Yulan Yan, Bhuvan Middha, and Xing Xie. 2021. Reinforcing pretrained models for generating attractive text advertisements. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3697–3707.

Penghui Wei, Xuanhua Yang, Shaoguo Liu, Liang Wang, and Bo Zheng. 2022. Creator: Ctr-driven advertising text generation with controlled pre-training and contrastive fine-tuning. *arXiv preprint arXiv:2205.08943*.

Sean Welleck, Ilya Kulikov, Stephen Roller, Emily Dinnan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *ICLR*.

Chao Zhang, Jingbo Zhou, Xiaoling Zang, Qing Xu, Liang Yin, Xiang He, Lin Liu, Haoyi Xiong, and Dejing Dou. 2021a. Chase: Commonsense-enriched advertising on search engine with explicit knowledge. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4352–4361.

Xueying Zhang, Yanyan Zou, Hainan Zhang, Jing Zhou, Shiliang Diao, Jijia Chen, Zhuoye Ding, Zhen He, Xueqi He, Yun Xiao, et al. 2021b. Automatic product copywriting for e-commerce. *arXiv preprint arXiv:2112.11915*.

A Appendix

A.1 Post-Pretraining Datasets

We collect all post-pretraining datasets from a Chinese leading advertising platform. Specifically,

- **Datasets for Ad Description Generation** We collect tens of millions of <search ad title, search ad description> pairs manually written by advertisers from the leading advertising platform as inputs and outputs of ad description task. For each pair, we match corresponding product landing page with it as auxiliary attributes.

- **Datasets for Ad Title Generation** For the ad title generation task, we use product entities as input text, corresponding product landing pages as auxiliary attributes and feed ad titles human-written by advertisers from a leading advertising platform as output text.

- **Datasets for Selling Point Generation** We construct selling point generation datasets by extracting snippets related to product attributes in the above ad descriptions and ad title dataset as product selling points via public universal information extraction tools ⁴.

- **Datasets for Comment Generation** We use hard prompt methods (Brown et al., 2020; Schick and Schütze, 2020) to obtain datasets for ad comment generation. Specifically, we use product descriptions and human-written prompt templates (e.g., what do you think about this product) as the context of the PLATO-Ad model in the pretraining phase to generate generic-domain comments (responses). We use publicly-available text sentiment analysis tools ⁵ to filter out negative text.

- **Datasets for Open-domain QA** We collect open-domain QA datasets from the Chinese community-based question-answering websites (just like Quora ⁶) in the leading advertising platform. Here the question and answer of each QA item are treated as input text and output text respectively. Meanwhile, we use publicly available named entity recognition tools ⁷ to extract the entities of questions as auxiliary keywords (Attributes).

For all collected data, we will conduct a data-preprocess procedure, including filtering out low-quality ones via a set of heuristic rules and publicly-available tools (e.g., text error detection ⁸, sentence length constraint, repeat word constraint and harmful/abusive word vocabulary).

⁴https://github.com/PaddlePaddle/PaddleNLP/tree/develop/model_zoo/uie

⁵https://ai.baidu.com/tech/nlp_apply/sentiment_classify

⁶<https://www.quora.com/>

⁷https://ai.baidu.com/tech/nlp_basic/entity_analysis

⁸https://ai.baidu.com/tech/nlp_apply/text_corrector

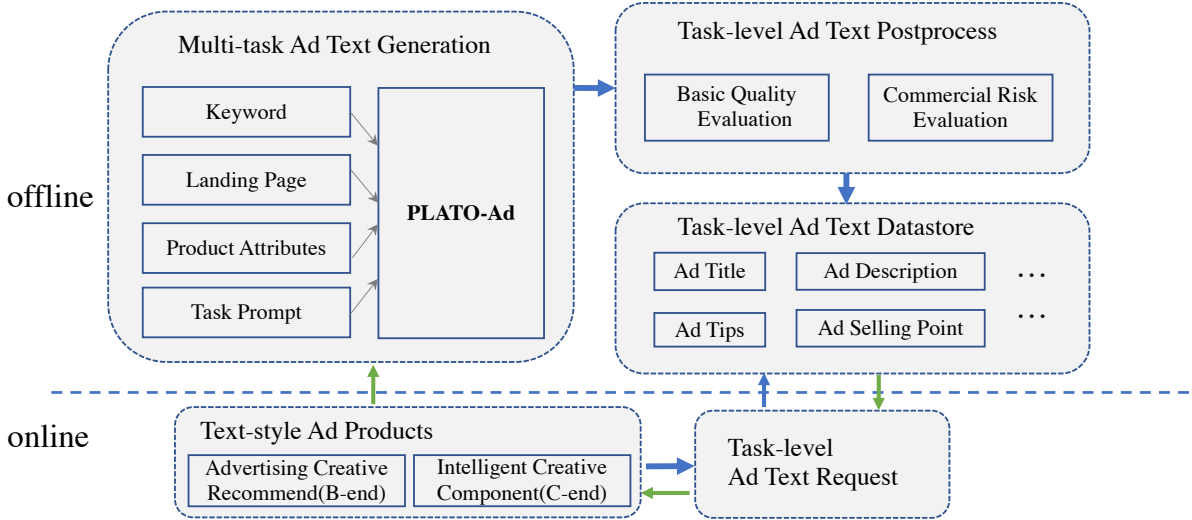


Figure 4: Deployment workflow of PLATO-Ad.

A.2 Model Settings

Our PLATO-Ad is a 93M parameter model with 12 transformer blocks and 12 attention heads, with the embedding dimension of 768. The model structure follows the setting of PLATO-2 (Bao et al., 2021). According to the number of post-pretraining tasks, we set five kinds of task prompts and the embedding size of task prompt is set to be 768. We randomly initialize the representation vectors of task prompts. We train PLATO-Ad in post-pretraining phase with the batch size of 65,536 on 8 A100 GPUs. During decoding, we adopt a topk-sampling decoding strategy with $k=5$. $\lambda_u, \lambda_r, \lambda_c$ are set to be 1.

A.3 Automatic Evaluation

Perplexity (Brown et al., 1992) is an evaluation metric to measure the model capacity for language modeling which is the normalized inverse probability of the dataset. BLEU is to use n-gram word matching to measure the similarity between golden truth and generated text. In this paper, we use Pairwise-BLEU (Shen et al., 2019) to measure the diversity of generation results. Specifically, Pairwise-BLEU measures similarity among the hypotheses (multiple generated candidate results). The more diverse the hypothesis set is, the lower the Pairwise-BLEU is.

A.4 Manual Evaluation

We conduct a manual evaluation on 200 random samples from our test dataset. Three participants were recruited to measure the quality of the result generated by each baseline from three perspectives.

Each perspective is measured by a 3-point Likert question where 0 is bad, 1 is neutral and 2 is good.

- **Readability (Read.):** measures how the generated ad text is smooth and grammatically corrects.
- **Relevance (Rele.):** measure whether output text is relative with input text and whether the generated result is consistent with auxiliary attributes.
- **Information (Info.):** measures how informative/knowledgable generated ad text is.

Overall (Over.): measures the overall quality of generated ad texts, which is calculated by the average of the above three scores.

A.5 Deployment Workflow

Figure 4 shows the deployment workflow of PLATO-Ad. We first use PLATO-Ad to generate ads offline. Then, before online deployment, we conduct a task-level ad text post-processing procedure for all generated advertisements, including the basic quality evaluation (quality control) and commercial risk control evaluation to strictly control the suitability of displayed advertisements. Finally we store the filtered ads for online retrieval. Overall, the suitability rate of ads generated by PLATO-Ad is around 91% to 96%, which means that PLATO-Ad can be deployed online in the industry.

Meanwhile, for badcases or harmful contents that are found or fed back from customers when displayed online, we also have an online blacklist procedure to filter out them in real time.

Dense Feature Memory Augmented Transformers for COVID-19 Vaccination Search Classification

Jai Gupta Google Research jaigupta@google.com	Yi Tay Google Research yitay@google.com	Chaitanya Kamath Google Research ckamath@google.com	Vinh Q. Tran Google Research vqtran@google.com
Donald Metzler Google Research metzler@google.com	Shailesh Bavadekar Google Research shaileshb@google.com	Mimi Sun Google Research mimisun@google.com	Evgeniy Gabrilovich Google Research gabr@acm.org

Abstract

With the devastating outbreak of COVID-19, vaccines are one of the crucial lines of defense against mass infection in this global pandemic. Given the protection they provide, vaccines are becoming mandatory in certain social and professional settings. This paper presents a classification model for detecting COVID-19 vaccination related search queries, a machine learning model that is used to generate search insights for COVID-19 vaccinations. The proposed method combines and leverages advancements from modern state-of-the-art (SOTA) natural language understanding (NLU) techniques such as pretrained Transformers with traditional dense features. We propose a novel approach of considering dense features as memory tokens that the model can attend to. We show that this new modeling approach enables a significant improvement to the Vaccine Search Insights (VSI) task, improving a strong well-established gradient-boosting baseline by relative +15% improvement in F1 score and +14% in precision.

1 Introduction

Though COVID-19 continues to be a challenge worldwide, vaccines have provided the much needed hope. Countries and governments have significantly ramped up their efforts to improve the reach of the vaccines including booster shots. As such, it is important to understand how users search for vaccine related information such as vaccine efficacy, safety, and regional availability as this information can be very useful to inform policy decision making, create effective public service announcements, implement more efficient distribution of vaccines, etc. To this end, we released a public tool for COVID-19 Vaccination Search Insights, an interactive report that provides insights on user searches for COVID-19 vaccinations. For an example please see Figure 1. On the backend, this tool performs privacy preserving classification of

user search queries and clusters them based on the region they were issued from to present a timeline of how these searches have changed over time. A core task in the VSI tool is the problem of classifying search query intent, and one of them is whether users are seeking information on *vaccine access*, i.e. queries related to the eligibility, availability, and accessibility of COVID-19 vaccines.

The task is a challenging one since simply matching for COVID related terms is insufficient, as queries such as *fully vaccinated travel* or *proof of covid vaccination* are negative classes. Hence, this problem is nuanced and may benefit from a coalition of advanced language understanding systems and traditional search-related feature engineering methods. As such, the problem at hand crosses two main modalities, i.e., text and traditional dense features. In this problem, the text features are user search queries that are short and/or lack context. The dense features (discussed in more detail in section 5) are hand crafted features from recent and past activities, previous clicks, and named entities that play a critical role in understanding the user’s intent. However, dense features alone fail to capture important contextual language cues, such as those that state-of-the-art natural language understanding systems (Devlin et al., 2018; Raffel et al., 2019) have been shown to handle well. We find that both modalities are highly complementary and it is difficult to achieve strong performance using only a single modality.

Our Contributions The overall contributions of this paper can be summarized as follows:

- We propose a new model and framework for search query intent classification for our COVID-19 Vaccination Search Insight tool.
- We propose a paradigm of exploiting the benefits of text inputs through state-of-the-art NLU models, along with traditional dense features found in large-scale systems.

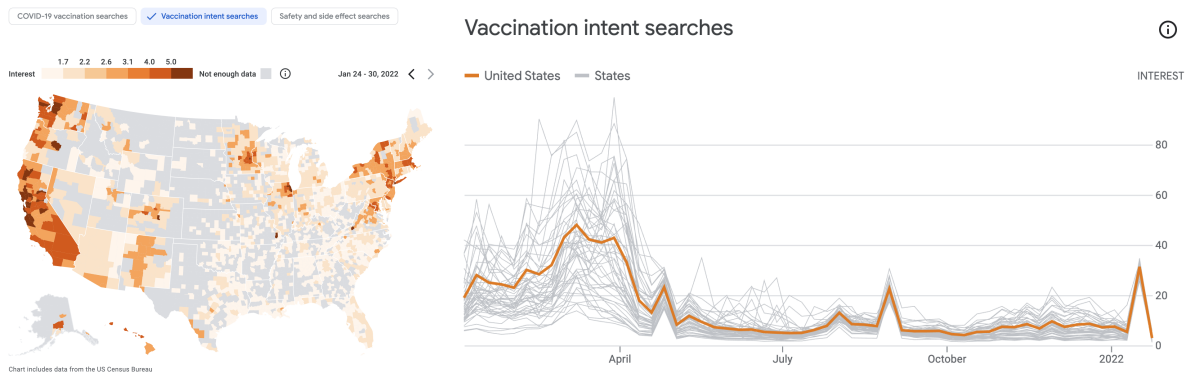


Figure 1: VSI tool presenting vaccination intent search query statistics segmented by location and time respectively.

- We propose a novel method of fusing dense features with Transformers that enables queries to retrieve from a dense memory store, in similar spirit to a contextual key-value store. Memory tokens here are used in similar manner as memory-based methods described in (Tay et al., 2020) and models such as Set Transformer (Lee et al., 2019), Memory Transformers (Sukhbaatar et al., 2019; Burtsev et al., 2020; Wu et al., 2020), and Global Memory tokens in ETC (Ravula et al., 2020) and BigBird (Zaheer et al., 2020). Notably, this is the first proposal for constructing global memory tokens using dense features.
- We conduct extensive experiments on real production data from three geographical regions. Our experiments show that the proposed method significantly outperforms a strong gradient boosting baseline by +15% and outperforms a SOTA Transformer by +5% on F1 score achieving very strong F1 score of over 98% on the US dataset with similar strong performance for other regions.

2 Related Works

This section presents related works and the background for this paper.

Classification with Feature-based ML. Building machine learned (ML) systems that operate across dense hand-crafted features is a well-established method. It is common to consider this class of ML methods as *tabular* machine learning whereby tree-based methods (Chen and Guestrin, 2016; Wikipedia, 2022) are dominant¹. Within the context of text classifiers, feature engineering typically

¹<https://www.kaggle.com/shivamb/data-science-trends-on-kaggle>

leverages stemming, lemmatization, part-of-speech tags, tf-idf vectors, entities, salient terms, and other features that are relevant to the task at hand. It is also popular to use semantic representations from Glove (Pennington et al., 2014) or BERT (Devlin et al., 2018) as input dense features to a model.

NLU with Pretrained Transformers. Transformers (Ashish Vaswani, 2017), characterized by interleaved self-attention and MLP blocks, have become the dominant sequence model for language processing and understanding (NLU) (Devlin et al., 2018; Raffel et al., 2019; Brown et al., 2020). The key idea behind self-attention is to perform token-to-token alignment where the joint interaction of queries and keys retrieve from a memory store (value). To this end, it is also common for advanced Transformer architectures to leverage global memory tokens (Zaheer et al., 2020; Lee et al., 2019; Jaegle et al., 2021) that act as a parameter store for the query to attend to (Tay et al., 2020). A cornerstone of these systems is the pretraining task that learns general purpose language representations, which have been shown to be extremely beneficial (Tay et al., 2021) due to the gains from transfer learning (Pan and Yang, 2010).

Joint Learning of Textual and Dense Features. Performing feature extraction to convert text into dense features (e.g. TF-IDF (Salton and Buckley, 1988), word2vec (Mikolov et al., 2013), etc.) for the purpose of learning classifiers jointly with other non-textual (numerical) dense features has been common practice in machine learning for some time (Kowsari et al., 2019; Macskassy et al., 1999; Richardson et al., 2007). Previous research in the multi-modal domain adopt a strategy of *early fusion* or *late fusion*: joining the two modalities (in this case textual and numerical) either in feature space early in the architecture or in seman-

tic/decision space late in the architecture, respectively (Snoek et al., 2005). Perhaps most related to our approach is a recent work on joint representation of text and tabular data (Yin et al., 2020; Zhu et al., 2021) that pass a flattened representation of a table alongside text during encoding. Our work instead provides a method for jointly encoding text with dense features of a more generic, unstructured form.

COVID-19 Vaccine Search Insights. An important distinction of Query classification from generic text classification is that the former are significantly shorter, and may be underspecified (He et al., 2000; Beitzel et al., 2005). Due to this nature, some previous works in this area have augmented queries with additional context to improve query classification performance (Broder et al., 2007; Shen et al., 2006; Li et al., 2008; Jiang).

Analyzing user queries and social interactions in health settings has been well studied. Sadilek et al. (2020) presents an analysis of user search queries for Lyme disease forecasting. Similarly, Sadilek et al. (2018) uses a machine-learned model for real-time detection of foodborne illness using web search and location data.

3 Problem Description

Given a search query q , the objective is to classify whether the query was issued with the intent of seeking information related to *vaccine access*. Notably, q is short in length and may or may not contain all the information needed to make the correct prediction.

Additionally, each q is supplemented with numerical features in the form of a dense feature vector $X_f \in \mathbb{R}^{d_{features}}$, where $d_{features}$ can be any number of features. Details about how X_f is constructed for our setup is present in section 5, but in short X_f represents the topicality scores of phrases related to the query with $d_{features}$ being 60k. However, it is important to note that all methods described in this paper are agnostic to the source of X_f and can be applied to any vector of numerical features.

4 VSI Transformer

This section describes the proposed method.

4.1 Pretrained Transformer Encoder

The main backbone of the proposed architecture is a Transformer (Ashish Vaswani, 2017) encoder.

We leverage the state-of-the-art T5 (Raffel et al., 2019) model as a starting point. Since T5 is a seq2seq model, we only utilize the T5 encoder as the Transformer model and discard the decoder of the pretrained model for our classification task. This is done by pooling the output of the encoder stack followed by a dense classification layer.

4.2 Input formulation

Given a query q , the input to the model is a discrete integer sequence representing the tokens (Google, 2021) of q , i.e., X_ℓ where ℓ is the number of tokens in the query from the subword vocabulary V . The input sequence is selected from an embedding matrix of $\mathbb{R}^{|V| \times d_{model}}$ to form a tensor of $\mathbb{R}^{\ell \times d_{model}}$.

4.3 Dense Feature Memory

For each input-target example, the input to the Dense Feature Memory module is a dense feature $X_f \in \mathbb{R}^{d_{features}}$. Given this dense feature of dimensions $d_{features}$, we transform it into memory tokens of dimension d_{model} via:

$$M_i = ReLU(W_i X_f + b_i)$$

where $W_i \in \mathbb{R}^{d_{features} \times d_{model}}$ and M_i is the i -th memory token. We consider the number of memory tokens to be a hyperparameter. To this end, we then concatenate $[M_1; \dots; M_{N_{memory}}]$ to the input query sequence $X \in \mathbb{R}^{\ell \times d_{model}}$. Along with the input query, we also pass in dense features corresponding to the query to the main body of the network. We note that memory tokens participate in the rest of the computation in a similar spirit to query tokens, i.e., they go through the same MLP and self-attention layers. The dense features are of $d_{features}$ dimensions and are passed into the dense memory module. The dense features used in our setup is explained in section 5.

4.4 Attention Blocks: Querying & Retrieving from Dense Feature Memory Tokens

The dense feature memory token is appended to the input sequence and participates in the self-attention mechanism of the Transformer model. Concretely, the QK matrix of the Transformer can be now written as:

$$A_{\ell,h} = Softmax([Q_{\ell,h}; m_{\ell,h}][K_{\ell,h}; m_{\ell,h}]_{\ell,h}^\top)$$

$$Y_{\ell,h} = A_{\ell,h}[V_{\ell,h}; v_{\ell,h}m_{\ell,h}]$$

where $Y_{\ell,h}$ is the h -th head of the output at layer ℓ , Q, K, V are the standard transformations of the

query input sequence, and $m_{\ell,h}$ is the dense feature memory token for layer ℓ . Since Q and V are both augmented with dense features, this provides an opportunity for both the dense features to align with query tokens and vice versa.

4.5 Output layer and Optimization

The final output layer of the Transformer stack is then passed into a pooling and MLP layer.

$$Y_{out} = MLP(\psi(Y_L)) \quad (1)$$

where $\psi(\cdot)$ is a pooling operator that maps $\mathbb{R}^{n \times d_{model}} \rightarrow \mathbb{R}^{d_{model}}$. Our $MLP(\cdot)$ function maps $\mathbb{R}^{d_{model}} \rightarrow \mathbb{R}^{N_{class}}$ to the number of classes. Our model then optimizes the Softmax cross entropy loss between the true classes and the predicted values. $L = \sum^L \sum_{i=1}^n y_i \log(\pi_i) + (1 - y_i) \log(1 - \pi_i)$, where π_i is the prediction of class i and y_i is the ground truth label of the class i .

5 COVID-19 Vaccine Access Dataset

Due to the novel nature of COVID-19, no previous datasets exist to accurately learn a model for the purpose of vaccine access query classification. Thus, in this section we outline the process we used to create this dataset.

Collection. To collect a dataset of queries to be labeled for vaccine access, we sample anonymized queries from real search traffic. Since a small minority of our search queries are for COVID-19 vaccination topics, we leveraged Google’s Knowledge Graph entities to find queries that included high confidence positives, potential positives, and close negatives. For example, for high precision candidates we sample top and random queries associated with the entity “COVID-19 Vaccination”, while for high recall low precision candidates, we sample queries that are only associated with the entity “COVID-19” or with “Vaccination”.

Labeling. To label this dataset for the specific purpose of vaccine access, we rely on a large pool of search quality raters who have deep experience with how health-related information needs are reflected in search queries. These raters were unknown to and independent of the developers of the classifiers. Each query is rated by three independent raters.

Label Expansion. We expand our dataset using label propagation to queries that are very similar to labeled queries. We include examples of positive and negative vaccine access queries in Table 5.

Dense Feature Augmentation. We augment our dataset by supplementing each query using dense features. To generate these dense features we use a combination of (1) entities mentioned in the query via a proprietary library analogous to Google Cloud Entity Analysis (goo, 2019) and (2) related search queries determined by a proprietary algorithm. We pool these two sources and use the 60,000 most common words and phrases to create a dense feature representation with dimension 60,000. At each dimension, we assign a relevance score for the phrase. For mentioned entities, this is analogous to salience in the Google Cloud Entity Analysis API. Table 6 shows some of the top features that are generated for each classification split.

6 Experiments

Below, we dive into experiment setups and results.

6.1 Baselines

We compare our proposed approach with three competitive baselines. The choice of baselines serves two primary purposes, i.e., (1) to show our method is competitive against well-established methods, and (2) to confirm certain scientific hypothesis by ablation-like studies. Please see Section A.1 in the Appendix for further implementation details on how we configure and train our models.

Adaboost A technique used to create an ensemble of weak learners that begins by fitting an estimator on the original dataset and then repeatedly fits additional estimators focusing more on examples that are misclassified by the combination of all the existing set of estimators. Mathematically, the ensembled AdaBoost classifier can be represented as: $F_N(x) = \sum_{n=1}^N f_n(x)$ which consists of N weak learners ($f_n(x)$) that are combined to create the ensemble model $F_N(x)$.

Query-only Transformer This baseline, simply a VSI Transformer without any dense features, is added to evaluate the upper bound of a language only state-of-the-art classifier.

Late Fusion Transformer This is an ablative baseline for the VSI Transformer. Instead of employing dense feature memory, we combine the dense features with the Transformer output at the final layers. Hence, we call this baseline *Late Fusion*, representing how the fusion of modalities is done at the final stages. Concretely, we concatenate the dense features to the pooled output from the transformer layer stack and then add a few layers

of MLP before adding the classification head. See figure 2 for setup details.

6.2 Results & Analysis

Table 2 presents the F1 and precision metrics from the vaccine intent classification tasks. VSI Transformer outperforms Adaboost models by relative +15.1% gain on the US locale, +17.7% gain on the CA locale, and +7.6% gain on the GB locale on F1 metric. The gains against traditionally strong ML models are substantial and compelling. When compared with NLU-only approaches (e.g., query-only Transformer), VSI Transformer again strongly outperforms the baseline. Finally, there are modest (but consistently strong) gains against the best and strongest baseline considered of up to +2.1% F1 score.

6.2.1 Importance of Text and Dense Features

We study text-only NLU models and feature-only state-of-the-art ML based Adaboost models. Generally, it is not clear if NLU-only models outperform Adaboost models (or vice versa). Both modalities have their fair share of wins and loses across the three datasets and six metrics. To this end, we find that the well-established version of combining text and dense feature to outperform both Adaboost and the Query-only Transformer, signifying the importance of having both modalities for building a successful model.

6.2.2 Dense Feature Memory vs Late Fusion

Late fusion is a well-established way to combine end-to-end deep learning with real world tabular features (Severyn and Moschitti, 2015; Tay et al., 2017). Our results show that, while the Transformer with Late Fusion performs the best out of all baselines, the VSI Transformer still comfortably outperforms the Late fusion method, whereby we show that our proposed integration is more effective. In regards to the problem space and domain, this also seems to imply that a deeper fusion of text and dense features can be key in obtaining better model quality.

6.2.3 Increasing depth of MLP network

The experiments described above used a single layer in the MLP networks in both the architectures present in Figure 2. Adding a single layer means that we have almost the same number of additional parameters added in both the architectures for a fair comparison.

In this ablation, we study the impact of using multiple layers in the MLP network. Note that each of these layers has a dimension size of 768 and uses GeLU activation function. We see a consistent increase in performance as we increase the number of layers in the MLP network. This increase is more prominent in the Late Fusion architecture which starts to catch up (at the cost of increasing the depth of the model) but still performs worse than the VSI Transformer indicating that for the same number of model parameters, VSI Transformer is a better architecture.

6.2.4 Using multiple memory tokens

Table 3 presents the results on increasing the number of memory tokens (N_{memory}) to up to 4 tokens. Though intuitively, it might seem that the performance will improve, experiments show that the correlation is not that straightforward.

The metrics seem to improve slightly but there is a consistent degradation observed with the F1 metric as we increase N_{memory} to 4 tokens. When increasing N_{memory} from 3 to 4 tokens, all three regions show a degradation. Given the sequence length is constant, our hypothesis is that as we increase the number of memory tokens allocated to the dense features, we are using up tokens that could have otherwise been allocated to the query text. This reduces the length of query that can be seen by the model for large queries thereby impacting the overall performance of the model. Hence, in general, a single memory token might be sufficient, but the optimal number might differ from task to task.

6.2.5 Analyzing improvement patterns

Given the improvement in metrics, we looked at some of the examples where VSI Transformer model provides large gains over a query only model. An example pattern is "covid vaccine" followed by some noun. If the noun represents a location, the intent is to search for vaccine availability in that region. The query only Transformer model can only guess whether this represents a location unless it can recall from locations memoized during pretraining. Given that VSI transformer also uses dense features, it has additional signals that provide information like whether the query includes a location by sources like Google Cloud Entity analysis. Hence, using this additional signal, the model is able to cut down on a lot of false positives of this pattern.

Model	Input Features	US	CA	GB
Query-only Transformer	Q only	0.9395 / 0.9060	0.8715 / 0.8059	0.8896 / 0.8159
AdaBoost 20 estimators	DF only	0.8288 / 0.8399	0.7830 / 0.7918	0.8909 / 0.9019
AdaBoost 50 estimators	DF only	0.8570 / 0.8598	0.8315 / 0.8678	0.9132 / 0.9128
Transformer Late Fusion	Q + DF	0.9780 / 0.9698	0.9585 / 0.9289	0.9719 / 0.9519
VSI Transformer	Q + DF	0.9868 / 0.9809	0.9784 / 0.9655	0.9824 / 0.9730
% Improvement (vs Query-only)	-	+5.0% / 8.3%	+12.3% / +19.8%	+10.4% / +19.3%
% Improvement (vs Adaboost)	-	+15.1% / +14.1%	+17.7% / +11.3%	+7.6% / +6.6%
% Improvement (vs best)	-	+0.8% / +1.1%	+2.1% / +3.9%	+1.1% / +2.2%

Table 1: F1 and Precision metrics on COVID-19 vaccination access search intent prediction. VSI Transformer outperforms best Transformer baseline by +0.8% to +2.1% and Adaboost by up to +17.7% F1 score.

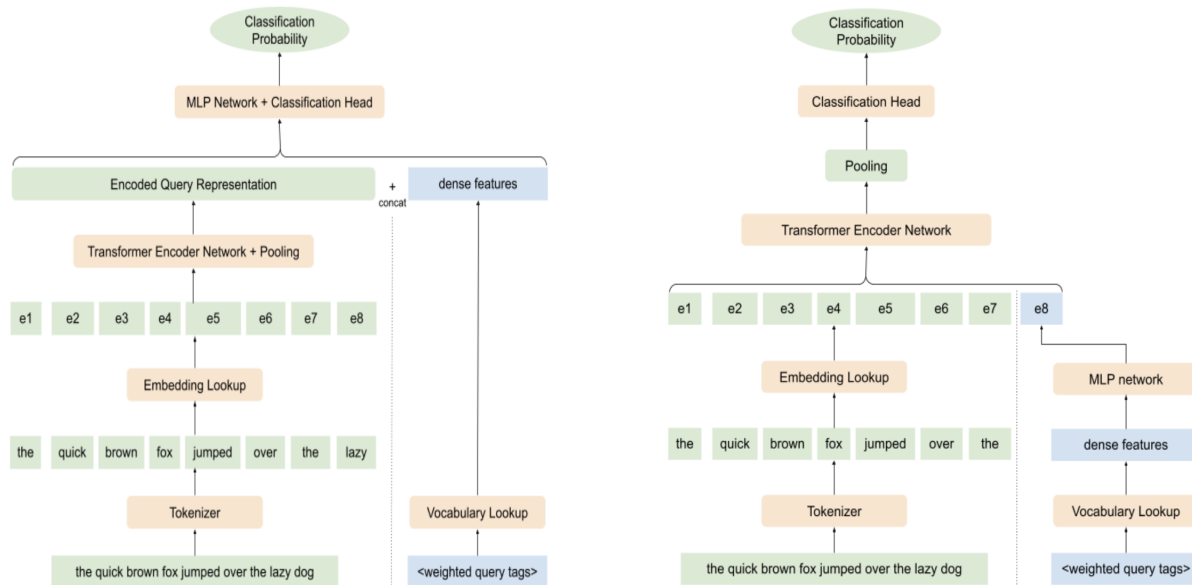


Figure 2: Depiction of adding transformed dense features to Transformer based models for a sample query. Architecture on the left depicts late fusion of the dense features to the query embeddings while the one on the right depicts addition of dense features to the query embeddings as a single memory token. Assumes that both the models have sequence length of 8 and the tokenizer produces one token for every word.

Model	N_ℓ	US	CA	GB
Late Fusion	1	0.9780	0.9585	0.9719
Late Fusion	2	0.9827	0.9690	0.9817
Late Fusion	3	0.9846	0.9788	0.9822
VSI	1	0.9868	0.9784	0.9824
VSI	2	0.9870	0.9785	0.9840
VSI	3	0.9872	0.9789	0.9848

Table 2: Impact of increasing the number of layers (N_ℓ) of the MLP networks on F1.

6.2.6 Using smaller Transformer models

The above experiments are performed with T5 1.1 Base model, but models of this size are often prohibitive for online applications due to their resource requirements and latency where smaller and shallower models are preferred. But shallower models tend not to perform as good and One possible reason is that shallower models are limited in their capability to extract complex features.

N_{memory}	US	CA	GB
0	0.9395	0.8715	0.8896
1	0.9868	0.9784	0.9824
2	0.9875	0.9784	0.9850
3	0.9871	0.9788	0.9849
4	0.9853	0.9785	0.9839

Table 3: Trend of F1 metric on increasing the number of chosen memory tokens N_{memory} in VSI transformer architecture.

Given that dense features can provide complex features as inputs to the model in a preprocessed format, using dense features can provide a large boost in quality for shallow models. Additionally, such features can provide exclusive information that is not available in the text features (query). Hence, we performed an ablation study on usefulness of dense features on the model size.

Table 4 presents results on running the same experiments as above, but using T5 1.1 Small instead of the Base model. Overall, we see larger relative improvements compared to the Base model. For example, the VSI Transformer Small model has a relative F1 gain of 6.0% and precision gain of 10.4% over query-only Transformer Small model, much higher than the relative F1 gain of 5% and precision gain of 8.3% observed with the Base model.

Model	Small	Base
Query-only	0.9308 / 0.8866	0.9395 / 0.9060
Late Fusion	0.9539/0.9336	0.9780 / 0.9698
% Imp. (vs Query-only)	+2.5%/+5.3%	+4.1% / 7.0%
VSI Transformer	0.9862/0.9788	0.9868 / 0.9809
% Imp. (vs Query-only)	+6.0%/+10.4%	+5.0% / 8.3%

Table 4: Comparison of F1 and precision metric between T5 1.1 small and base models on US dataset.

Even through shallow, the VSI Transformer’s architecture is able to make better use of the attention layers for the dense features leading to pretty high boost even over the Late Fusion architecture, affirming that it is a better architecture for shallow models as well.

7 Limitations

Though we have shown that just one memory token is sufficient, assigning tokens to dense features means less number of tokens are available in the sequence for the text input. Another limitation is the use of locale specific vocabularies for dense features for each regional dataset as present in table 5, but that is not a limitation of the VSI Transformer but instead how the dense features are generated.

8 Conclusion

This paper presented an important task of classifying search queries with COVID-19 vaccination access intent. With an extensive set of experiments and comparing with strong baselines, we presented VSI Transformer, a novel and generic approach that consistently and strongly outperforms all existing baselines that operate on either of the two modalities, or late fusion of the modalities. With an ablation study on model size, we show that for online applications where shallower models need to be deployed primarily due to latency constraints, making use of dense features can help bridge the gap in performance compared to deeper models. Future work in this direction can help further understand how to choose the optimal number of memory to-

kens, and explore more architectures to efficiently combine sequential and dense features.

Ethics Statement

In our ongoing fight against the COVID-19 pandemic, understanding whether search queries exhibit an intent to seek vaccine access is an important problem to study to be able to collect search statistics about COVID-19 vaccination efforts. These statistics are made public via an online website, an interactive report that is accessible by anyone. All data used to train these models are anonymized and sampled, and labeled by a large pool of search quality raters who are trained to assess health-related information needs in search queries. These raters are unknown to and independent from the developers of the classifiers. This dataset is not made public or used in any other context.

References

2019. Google Cloud natural language api basics. ["https://cloud.google.com/natural-language/docs/basics#entity_analysis](https://cloud.google.com/natural-language/docs/basics#entity_analysis).
- Niki Parmar Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. 2017. *Attention is all you need*. 31st Conference on Neural Information Processing Systems (NIPS).
- Steven M. Beitzel, Eric C. Jensen, Ophir Frieder, David A. Grossman, David D. Lewis, Abdur Chowdhury, and Aleksander Kolcz. 2005. Automatic web query classification using labeled and unlabeled training data. In *SIGIR '05*.
- Andrei Z. Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. [Robust classification of rare queries using web knowledge](#). In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, page 231–238, New York, NY, USA. Association for Computing Machinery.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Mikhail S. Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V. Sapunov. 2020. [Memory transformer](#).

- 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.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Github Google. 2021. [Sentencepiece Tokenizer](#).
- Daqing He, Daqing, Göker, and Ayse Goker. 2000. Detecting session boundaries from web user logs.
- Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. 2021. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*.
- Daxin Jiang () and Jian-Tao Sun. 2009. [Context-aware query classification](#). In *SIGIR' 09, The 32nd Annual ACM SIGIR Conference*. Association for Computing Machinery, Inc.
- Kowsari, Jafari Meimandi, Heidarysafa, Mendu, Barnes, and Brown. 2019. [Text classification algorithms: A survey](#). *Information*, 10(4):150.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosior, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR.
- Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. [Learning query intent from regularized click graphs](#). In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, page 339–346, New York, NY, USA. Association for Computing Machinery.
- Sofus Macskassy, Aynur Dayanik, Haym Hirsh, and Morrocroft Ii. 1999. Emailvalet: Learning user preferences for wireless email.
- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. [Efficient estimation of word representations in vector space](#).
- Sinno Jialin Pan and Qiang Yang. 2010. [A survey on transfer learning](#). *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.
- 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.
- 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*.
- Anirudh Ravula, Chris Alberti, Joshua Ainslie, Li Yang, Philip Minh Pham, Qifan Wang, Santiago Ontanon, Sumit Kumar Sanghai, Vaclav Cvicek, and Zach Fisher. 2020. [Etc: Encoding long and structured inputs in transformers](#). In *2020 Conference on Empirical Methods in Natural Language Processing (EMNLP 2020)*.
- Matthew Richardson, Ewa Dominowska, and Robert Ragno. 2007. [Predicting clicks: Estimating the click-through rate for new ads](#). In *Proceedings of the 16th International Conference on World Wide Web, WWW '07*, page 521–530, New York, NY, USA. Association for Computing Machinery.
- Adam Sadilek, Stephanie Caty, Lauren DiPrete, Raed Mansour, Tom Schenk, Mark Bergtholdt, Ashish Kumar Jha, Prem Ramaswami, and Evgeniy Gabrilovich. 2018. Machine-learned epidemiology: real-time detection of foodborne illness at scale. *NPJ Digital Medicine*, 1.
- Adam Sadilek, Yulin Hswen, Shailesh Bavadekar, Tomer Shekel, John Brownstein, and Evgeniy Gabrilovich. 2020. [Lymelight: forecasting lyme disease risk using web search data](#). *npj Digital Medicine*.
- Gerard Salton and Christopher Buckley. 1988. [Term-weighting approaches in automatic text retrieval](#). *Information Processing Management*, 24(5):513–523.
- Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 373–382.
- Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- Noam Shazeer, Youlong Cheng, Niki Parmar, Dustin Tran, Ashish Vaswani, Penporn Koanantakool, Peter Hawkins, HyoukJoong Lee, Mingsheng Hong, Cliff Young, et al. 2018. Mesh-tensorflow: Deep learning for supercomputers. *Advances in neural information processing systems*, 31.
- Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pages 4596–4604. PMLR.
- Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2006. [Building bridges for web query classification](#). In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06*, page 131–138, New York, NY, USA. Association for Computing Machinery.

Cees G. M. Snoek, Marcel Worring, and Arnold W. M. Smeulders. 2005. [Early versus late fusion in semantic video analysis](#). In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, MULTIMEDIA '05, page 399–402, New York, NY, USA. Association for Computing Machinery.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Yi Tay, Mostafa Dehghani, Jai Gupta, Dara Bahri, Vamsi Aribandi, Zhen Qin, and Donald Metzler. 2021. Are pre-trained convolutions better than pre-trained transformers? *arXiv preprint arXiv:2105.03322*.

Yi Tay, Minh C Phan, Luu Anh Tuan, and Siu Cheung Hui. 2017. Learning to rank question answer pairs with holographic dual lstm architecture. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 695–704.

Wikipedia. 2022. [AdaBoost](#).

Qingyang Wu, Zhenzhong Lan, Jing Gu, and Zhou Yu. 2020. [Memformer: The memory-augmented transformer](#).

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.

Manzil Zaheer, Guru Guruganesh, Kumar 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. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. [TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3277–3287, Online. Association for Computational Linguistics.

A Appendix

A.1 Implementation Details

The VSI Transformer is implemented in Mesh Tensorflow (Shazeer et al., 2018), a Tensorflow-like

API that supports distributed model parallelism. We initialize our model with the T5.1.1 Base checkpoint, comprised of 12 encoder layers, d_{model} size of 768, d_{ff} of 2048. The model uses GEGLU-based feed-forward layers as described in (Shazeer, 2020). The model has 12 heads. The overall number of non-embedding parameters is approximately 100M parameters. The model also utilizes the standard 32K SentencePiece that was trained on the C4 corpus. Our model is trained with 16 TPUv3 chips. We finetune all models using a sequence length of 32 subword tokens using the Adafactor optimizer (Shazeer and Stern, 2018). The learning rate is a constant learning rate of 10^{-3} and the batch size is 128.

A.2 Top Search Queries

Figure 3 presents a view of the tool listing top search queries associated with vaccination intent.

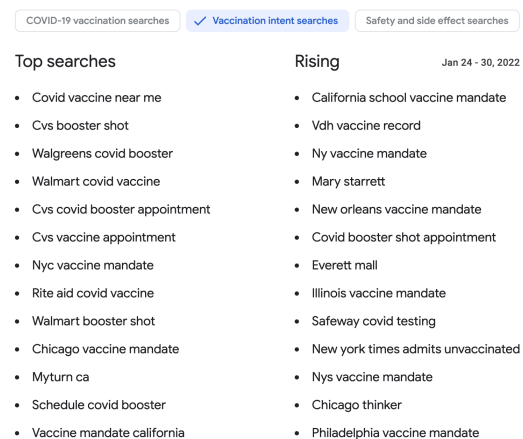


Figure 3: VSI tool presenting top search queries with vaccination intent

A.3 Sample Queries

Table 5 presents a sample of some positive and negative queries for vaccine acces present in our dataset. The examples in the table shows how nuanced the problem is and why simply looking for termsatches didn't perform as good and a more sophisticated approach was needed.

A.4 Dense Feature Vocabulary

As mention in section 5, the dataset consists of dense features created from a vocabulary of 60,000 phrases. Table 6 presents a list of top phrases present in the vocabulary of each of the 3 regions.

Class	Examples
Positive	<i>covid vaccine appointment where can i get covid vaccine book covid jab walk-in covid vaccine near me nhs book covid vaccine</i>
Negative	<i>covid stats by country covid vaccine effectiveness fully vaccinated travel proof of covid vaccination how long does the vaccine last</i>

Table 5: Examples of positive and negative COVID-19 vaccine access queries.

Category	Top Features
Vaccination access (US)	<i>pharmacy, pfizer, vaccine appointment, appointment, pharmacies, moderna, dose, appointments, pfizer vaccine, cvs, walgreens, second dose, vaccine appointments, cvs pharmacy, doses, shot, cvs covid, walgreens pharmacy, vaccine eligibility, moderna vaccine</i>
Vaccination access (GB)	<i>appointment, appointments, book, booking, vaccination centre, clinic, vaccination centres, vaccine appointment, clinics, nhs uk, nhs, coronavirus covid, walk in, coronavirus vaccination, covid vaccination, vaccination clinic, vaccine clinic, centres, vaccination appointment, vaccine centre, centre, book covid, pfizer, astrazeneca</i>
Vaccination access (CA)	<i>clinic, appointment, clinics, vaccine clinic, vaccine appointment, vaccination clinic, pharmacy, appointments, book, booking, pharmacies, registration, pfizer, drug mart, vaccination center, walk in, vaccination appointment, vaccination clinics, vaccine pop up</i>

Table 6: Top features associated with a query in the dense feature vector, for each split.

Full-Stack Information Extraction System for Cybersecurity Intelligence

Youngja Park and Taesung Lee

IBM T. J. Watson Research

Yorktown Heights, NY 10598, USA

young_park@us.ibm.com, taesung.lee@ibm.com

Abstract

Due to rapidly growing cyber-attacks and security vulnerabilities, many reports on cyber-threat intelligence (CTI) are being published daily. While these reports can help security analysts to understand on-going cyber threats, the overwhelming amount of information makes it difficult to digest the information in a timely manner. This paper presents, SecIE, an industrial-strength full-stack information extraction (IE) system for the security domain. SecIE can extract a large number of security entities, relations and the temporal information of the relations, which is critical for cyberthreat investigations. Our evaluation with 133 labeled threat reports containing 108,021 tokens shows that SecIE achieves over 92% F1-score for entity extraction and about 70% F1-score for relation extraction. We also showcase how SecIE can be used for downstream security applications.

1 Introduction

A rapid increase in cyberattacks, both in number and attack techniques, poses enormous challenges to security analysts. Much of the information on new threats often appear first in unstructured reports such as blogs and news articles. To quickly respond to the on-going attacks, it is critical to digest the information about new threats in a short period of time. However, it is very difficult to find relevant information from CTI reports, particularly because cyber-attacks involve many different entities, including the attacker, victim (e.g., companies/industries), tools (e.g., malware) indicators of compromise (IOCs, e.g., file names and IP addresses), and various relations, some of which may be unknown to the security experts.

We present a large-scale full-stack IE system designed for the cybersecurity domain. SecIE can extract 26 entity types, 20 fixed rela-

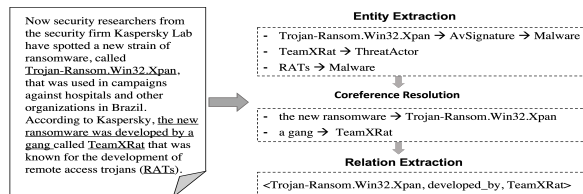


Figure 1: A CTI report and the security entities and relation extracted by SecIE

tion types and various Open IE relations, and the time information of the relations, which is very critical in cybersecurity. Figure 1 shows a snippet of a CTI report¹ and the IE results from SecIE. The entity extraction model detects mentions of *Malware* and *ThreatActor* from the text. The coreference resolution model recognizes that ‘the new ransomware’ refers to *Trojan-Ransom.Win32.Xpan* and ‘a gang’ refers to *TeamXRat*. Finally, the relation extraction module produces a relation tuple, $\langle \textit{Trojan-Ransom.Win32.Xpan}, \textit{developed_by}, \textit{TeamXRat} \rangle$, from “the new ransomware was developed by a gang”.

While there have been efforts to apply NLP and IE to the cybersecurity domain (Joshi et al., 2013; Lal, 2013; Jones et al., 2015; Bridges et al., 2017; Liao et al., 2016; Husari et al., 2017; Pingle et al., 2019; Yi et al., 2020), they target on a specific sub-area of cybersecurity, mostly on extracting IOCs or vulnerabilities, or a component (either entity extraction or relation classification) in the IE process. To our knowledge, our system is the largest end-to-end IE system for the cybersecurity domain supporting a large number of security entity and relation types.

Most existing IE systems apply supervised (deep) learning methods relying on a large

¹<https://www.cyberdefensemagazine.com/teamxrat-spreads-ransomware-via-rdp-brute-force-attacks/>

amount of high-quality labeled data. Unlike the general domain types, labeling fine-grained security entities and relations requires deep domain knowledge, and, thus it is much more difficult to produce a high-quality training data for the security domain. As an anecdote, 3 annotators (1 security expert and 2 professional annotators with many years’ experience) working full-time for 5 months could produce only 133 annotated documents, which are far from enough to train supervised models for our need. Thus, SecIE applies unsupervised NLP technologies. We develop techniques to handle idiosyncrasies in security terms and take into account the structural characteristics found in many CTI reports. This domain customization allows SecIE highly accurate, achieving over 92% F1 for entity extraction and 70% F1 for relation extraction.

2 Methodology

We employ a pipeline architecture as shown in Figure 2, consisting of document parsing; linguistic analysis; entity extraction; coreference resolution; topic entity detection; relation extraction; and relation time assignment. Input

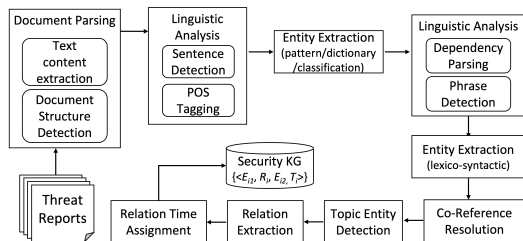


Figure 2: High-level System Architecture

documents are processed sequentially, where the document content and all the results from the previous components are passed as input to the next component. However, the system can process multiple documents in parallel yielding a high throughput.

2.1 Document Pre-processing

Document Parsing: This component performs text content extraction and document structure detection. We use Apache Tika² to extract the file content and structural information such as titles, hyperlinks, tables, and list structures from the input files. The extracted structures are stored as annotations over the

²<https://tika.apache.org>

document content and passed to the subsequent components along with the content.

Linguistic Analysis: This component performs sentence boundary detection, part-of-speech (POS) tagging and dependency parsing. We use SyntaxNet (Andor et al., 2016) for POS tagging and dependency parsing. It was trained with general domain documents and often fails to parse security sentences correctly, because some security entities include many tokens and punctuation marks internally (e.g., some URLs have over 100 tokens). To improve the parsing accuracy, we first detect entity mentions and pass the entire entity mention as a noun token to the parser. Figure 3 shows a sample sentence and the parsing results when all tokens are passed to the parser individually and when entity mentions are passed as a token.

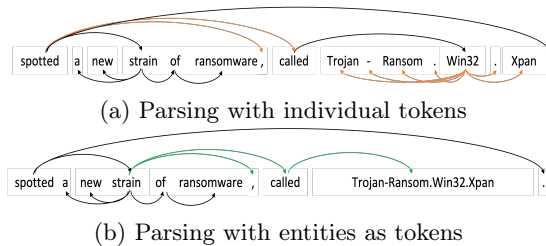


Figure 3: Improved sentence parsing through domain customization

2.2 Entity Extraction

We identified the 26 fine-grained entity types related to malware, IoCs, and security vulnerability. The types are determined based on the STIX standard³ which defines 9 key security concepts and their relationships. The full list of our entity types are shown in Figure 6 in Appendix. We provide a type inheritance as shown in Figure 6, allowing applications to consume the entity types at different levels.

Pattern-based Method is used for entity types with well-defined patterns such as *IPAddress* and *EmailAddress*. We note that many CTI reports, especially those published online, often use obfuscated forms for malware IOCs, such as ‘X.Y.177.245’, ‘82(dot)103(dot)137(dot)14’, ‘BLOCKED.BLOCKED.172.196’, and ‘x0x0.[REMOVED].com.br’. Thus, SecIE supports many obfuscated IOC patterns, unlike other existing tools.

³<https://oasis-open.github.io/cti-documentation/>

Dictionary-based Method is used when a reputable list of terms belonging to a certain entity type exists. In the cybersecurity domain, previously known *Campaign*, *Malware* and *ThreatActor* cases are well documented. In these cases, we match the dictionary terms with the noun phrases. However, this dictionary matching method can extract only previously known samples. We address this problem using the lexico-syntactic pattern matching method to extract new mentions.

Lexico-Syntactic Pattern-based Method

Inspired by the findings in (Hearst, 1992), we apply the following syntactic patterns to extract security entities: (1) NP (, NP)* BE NP; (2) NP, CALLED NP; (3) NP such as NP (, NP)* (4) NP including NP (, NP)*; (5) NP a.k.a | ((which|that)? (BE)? (also)? CALLED as) NP. Here, NP stands for a noun phrase, BE represents the be-verbs (e.g., ‘is’), and CALLED includes ‘dubbed’, ‘called’, ‘named’, ‘known’, ‘referred’, and ‘termed’.

To discover new mentions, we first check if the entity type of an NP in these syntactic patterns is determined. Then, we label the remaining NPs to the same entity type. If the types of multiple NPs are determined, they should be the same type or have a super-subtype relation. We also use a predefined set of cue words to detect new mentions for *Campaign* and *Malware*, and *ThreatActor* (Table 6 in Appendix). If a cue word matches with NP or NP’s headword, we classify the other NPs to the same entity type as the cue word. Table 1 shows sample sentences where ‘WannaCry’, ‘Wcrypt’, ‘WCRY’, ‘WannaCrypt’ are extracted as *Malware* even though the mentions were unknown.

-
- 1) WannaCry is a ransomware worm that spread rapidly ...
 - 2) A new ransomware dubbed "WannaCry" is ...
 - 3) The WannaCry ransomware has been very active since ...
 - 5) WannaCry, also known as Wcrypt, WCRY, WannaCrypt
-

Table 1: Examples of new mention extraction. The numbers indicate the rule used to determine the entity type.

Classification-based Extraction *AvSignature* mentions do not conform to particular patterns making regex-based method ineffective (e.g., ‘ADWARE/Agent.imv’, ‘Trojan-Ransom.Win32.Xpan’). Further, the number of *AvSignature* instances is very large (millions),

making the dictionary method inefficient. However, they have distinct word shapes which are very different from regular words, and it is easy to collect many examples from public sources. We collected 660,000 *AvSignature* names from *VirusTotal* as the positive sample and added 470,000 words randomly chosen from CTI reports as the negative sample. We then trained a Logistic Regression model using character n-gram and word shape features (e.g., uppercase/lowercase letters, digits and symbols).

2.3 Coreference Resolution

We categorize coreferences into two types based on the search range for the referent.

Within-sentence Coreference appears in certain syntactic structures that connect two noun phrases, such as appositives, relative pronouns (e.g., ‘which’), or certain phrases such as “<nominative noun> [,] CALLED [as] <proper noun>”. When the proper noun belongs to a security entity, we resolve the nominative noun or pronoun to the proper noun. Figure 1 shows two examples of within-sentence coreferences: “a new strain of ransomware, called Trojan-Ransom.Win32.Xpan” and “a gang called TeamXRat”. We resolve “a new strain of ransomware” to “Trojan-Ransom.Win32.Xpan” and “a gang” to “TeamXRat”.

Cross-sentence Coreference Syntactic analysis alone cannot connect two mentions together when they appear in different sentences. We use a document structure-based sentence embedding model proposed in (Lee and Park, 2019), which generates semantic representations for sentences using BERT (Devlin et al., 2019; Joshi et al., 2019). If a sentence contains a nominative or pronoun mention (e.g., ‘the malware’), we identify semantically related sentences for the sentence based on the sentence embeddings and find its referent from the proper nouns in the related sentences. We replace the nominative or pronoun mention with each of the candidates, calculate the likelihood of the candidate in the sentence, and choose the candidate with the highest likelihood as the referent.

2.4 Topic Entity Detection

Most CTI reports provide a deep analysis on a particular malware or campaign. We call the focus of a CTI report the topic entity. Many

CTI reports are very succinct, often simply providing the list of related entities, such as IOCs, without contextual connection to the topic entity. These related entities provide critical intelligence about the topic entity, and connecting them with the corresponding malware or campaign is critical. We identify the topic entity of CTI reports as follows. We first look for mentions of *Malware*, *Campaign*, and *ThreatActor* in the first 15 sentences. When there are multiple mentions of these types, we choose the topic entity based on the following factors: (1) the position of the sentence (likely to appear early in the article); (2) if the mention is a singular or plural (tend to be singular); (3) the syntactic role of the mention in the sentence (likely the subject or the object); (4) the occurrence count of the mention in the article (likely to appear many times).

2.5 Relation Extraction

Similarly to entity extraction, we apply several different techniques for relation extraction.

OpenIE Relation Extraction discovers relations from certain syntactic structures (Angeli et al., 2015; Banko et al., 2007; Soderland et al., 2010; Fader et al., 2011; Mausam et al., 2012; Roy et al., 2019). Many security relations involve actions (e.g., download, connect, etc.). Thus, we focus on the three syntactic structures containing a verb phrase and two noun phrases: $\langle \text{NP}(\text{subj})\text{-VP-NP}(\text{obj}) \rangle$, $\langle \text{NP-VPP-PP-NP} \rangle$, and $\langle \text{VP-NP-PP-NP} \rangle$, where pp is a preposition. We find these syntactic structures in sentences, and, if both NP arguments are security entity mentions, we extract a relation by associating the NPs with the VP as the relation type. Table 2 shows examples of semantic relations extracted using this method.

Cooccurrence-based Relation Extraction

While the OpenIE relations provide useful semantic relations, extracting relations only from the three structures can miss other relevant relations. We generate relations if two security entities co-occur in a sentence but are not connected by an OpenIE relation. The assumption is that if the two entities frequently appear together in the same sentence, they should be of interest to security analysts. We produce cooccurrence-based relations between the five main security entities: *Campaign*, *Indicator*,

Malware, *ThreatActor* and *Vulnerability* and assign a generic relation type ('related'). Table 3 shows sample co-occurrence-based relations.

Relations with Topic Entity As discussed above, many threat reports describe information about a particular security event or entity, and other entities in the document provide insights on the topic entity. In this work, if the entities in a list are not included in any other relations, we connect them to the topic entity via a relation type denoted as *related+EntityType* (e.g., relatedHash).

2.6 Temporal Information Extraction

Threat intelligence is time sensitive, and knowing when a security event has occurred is critical. Time information can be expressed in multiple ways, including point-in-time (e.g., "2016-05-25"), relative time (e.g., "last year"), time range (e.g., "2016–2017"), and embedded time (e.g., "CVE-2017-3018"). SecIE extracts these time expressions and normalize them to the timestamp. For relative time expressions, we infer their point-in-time based on an anchor time, which can be an absolute time expression in nearby sentences. If there are no point-in-time expressions in the document, we use the file's last modified time or the publication date as the anchor time. Then, we use the following priority orders to determine which temporal information gets assigned to a relation: (1) time in the same dependency construct; (2) time in the same sentence (3) time in the previous sentences; (4) the document's last modified time; and (5) The document's published time Figure 7 in Appendix shows a sample threat report and the output of SecIE including the entity, relation, and time information.

3 Performance Evaluation

To evaluate our system, we manually labeled 133 CTI reports, which contain 6,438 sentences and 108,021 tokens. The documents were labeled by 3 full time annotators over 5 months. To ensure the quality of the labeled data, we kept only the labels agreed by all 3 annotators, resulting in 3,295 entity and 1,216 relation mentions. More detailed statistics of the annotations are shown in Table 7 and Table 8 in Appendix.

Extracted Relation	Input Sentence
<Locky, spread_through, Necurs botnet>	Locky ransomware is again being spread through the Necurs botnet.
<zhCat, listen_on, port 1000>	If the attackers set up a zhCat instance listening on port 1000 on 192.168.116.128 ...
<zhCat, listen_on, 192.168.116.128>	
<Shamoon, delayed_on, Saudi Aramco>	the Iranians deployed the Shamoon malware on Saudi Aramco, ...

Table 2: Examples of OpenIE relation extraction

Extracted Relation	Input Sentence
<Dyre variants, related, win32k.sys>	New Dyre variants exploiting CVE-2015-0057, a use-after-free vulnerability in the win32k.sys component
<KaiXin EK, related, 125.77.31.181>	125.77.31.181 port 12113 - otc.szmshc.com:12113 - KaiXin EK
<KaiXin EK, related, otc.szmshc.com:12113>	

Table 3: Examples of occurrence-based relation extraction

3.1 Entity Extraction Results

Table 4 shows the performance of our entity extraction (see Table 9 and Table 10 in Appendix for the performance for all entity types). The evaluation is performed by measuring the mention-level precision (P), recall (R) and F1 scores over all entity types. SecIE_{all} reports the performance for all 133 reports, showing that SecIE achieves a very high F1 score with a good balance between precision and recall.

Further, we compare SecIE with a deep learning model to illustrate the challenges for applying supervised learning methods for cybersecurity data. We split the 133 labeled documents into train (80%), validation (10%) and test (10%) datasets, consisting of 106, 14 and 13 documents respectively, and trained a BERT model as described in (Devlin et al., 2019). The results (*small*) validate that SecIE significantly outperforms the BERT model.

Model	Precision	Recall	F1
SecIE _{all}	95.1	89.4	92.2
SecIE _{small}	89.8	84.7	87.2
BERT _{small}	83.3	70.3	76.2

Table 4: Performance of entity extraction models. *all* denotes the 133 labeled documents, and *small* denotes the 13 test dataset.

3.2 Relation Extraction Results

We measure the performance of relation extraction using four different settings.

- *ExactMatch*: An extracted relation and the ground truth must have the same entity spans, entity types and the relation type.
- *-eType*: The condition for the entity type match is removed from *ExactMatch*. This is mainly because *Malware* and *Campaign* are often interchangeably used.

- *-rType*: The condition for the relation type match is removed from *ExactMatch*.
- *-eType-rType*: Both the entity type and the relation type can be different.

Further, we evaluate the performance of relation extraction with and without co-reference resolution to show the effectiveness of the co-reference resolution step. Table 5 shows the evaluation results demonstrating SecIE’s effectiveness. It produces over 70% precision across all settings, and co-reference resolution improves the performance, especially the recall.

	Without Coref.			With Coref.		
	P	R	F1	P	R	F1
<i>ExactMatch</i>	70.5	65.0	67.6	70.1	65.5	67.7
<i>-eType</i>	72.7	66.9	69.7	72.6	67.8	70.2
<i>-rType</i>	72.9	65.6	69.1	72.3	66.0	69.0
<i>-eType-rType</i>	75.9	67.8	71.6	75.7	68.8	72.1

Table 5: Relation extraction performance using different matching strategies and coreference settings.

4 Security Applications

We demonstrate how SecIE can provide additional insights on security incidents.

4.1 Malware Analysis

SecIE can be used to build a knowledge graph (KG) on malware from text. Figure 4 shows an input document about WannaCry⁴ and the output KG. As we can see, SecIE extracted all of the security entities and connected them to the topic entity (*new variant of WannaCry*).

4.2 Inconsistency in CVEs

The NVD (national vulnerability database) provides information about known security vulnerabilities including the descriptions and asso-

⁴<https://www.cybereason.com/cybereason-reveals-a-new-variant-of-wannacry-ransomware/>

Cyberreason reveals a new variant of WannaCry ransomware
 Earlier today (May 13th) we have identified a new variant of WannaCry.

mssecsvc.exe
 Sha1: 6e37dd4ea21fd096b233161ec7af90c17b581638
 MD5: 73766565804dc0e56de6bf2574fcd3

tasksche.exe
 Sha1: 9b54c4c2fc77dc650d5446d2b1646cd5f45c99c8
 MD5: 71f4a163938478116734c724f8d5109e

As other variants of the recent WannaCry Ransomware attack, this variant is automatically executed by "Microsoft Security Center (2.0) Service" and is trying to spread by creating SMB connections to random IP addresses, both internal and external. In addition, we identified communication to the known C&C domain [www\[dot\]lucifer85d979\[dot\]iposd\[dot\]hgosuri\[dot\]sewrwergwea\[dot\]com](http://www[dot]lucifer85d979[dot]iposd[dot]hgosuri[dot]sewrwergwea[dot]com) in IPs 144.217.254.3 and 79.137.66.14.

<http://w3.research.IBM.com>
www.research.ibm.com/myPath1/myPath2
<http://79.137.66.14:80/testPath/testPath2>
<ftp://79.137.66.14/testpath.exe>

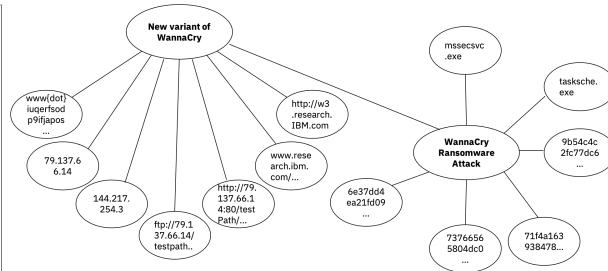


Figure 4: A KG built by SecIE from a report about the *WannaCry* ransomware

ciated metadata generated by domain experts. Even though the metadata was carefully curated by human, it can still contain errors. In particular, the affected software and the versions mentioned in the textual description and metadata can be different as shown in Figure 5. These inconsistencies can cause a harm, as many security applications rely on the metadata to identify vulnerable products in their environment.

Current Description	Known Affected Software Configurations
Buffer overflow in Adobe Acrobat 4.05 , Reader, Business Tools, and Fill In products that handle PDF files allows attackers to execute arbitrary commands via a long /Registry or /Ordering specifier.	<ul style="list-style-type: none"> * cpe:2.3:a:adobe:acrobat:3.0:*:*:*:*:* * cpe:2.3:a:adobe:acrobat:4.0:*:*:*:*:* * cpe:2.3:a:adobe:acrobat:4.0.5:*:*:*:*:* * cpe:2.3:a:adobe:acrobat_business_tools:4.0:*:*:*:*:* * cpe:2.3:a:adobe:acrobat_business_tools:4.0.5:*:*:*:*:* * cpe:2.3:a:adobe:acrobat_reader:3.0:*:*:*:*:* * cpe:2.3:a:adobe:acrobat_reader:4.0:*:*:*:*:* * cpe:2.3:a:adobe:acrobat_reader:4.0.5:*:*:*:*:*

Figure 5: Example of inconsistent CVE

We match the mentions of *Application* extracted from the description and the CPE entries in the metadata using simple matching rules. Since an application can be referred by several synonyms (*e.g.*, Microsoft Office vs. Office), we apply a loose matching for application names. The versions can be represented as an exact version (*e.g.*, 4.05), a range (*e.g.*, ‘before 10.3’), or wildcard symbols (*e.g.*, 4.x or 4.*), so we match the versions accordingly.

We randomly selected 168 CVE records and manually checked the inconsistency check results. This technique detected 26 potential inconsistencies, and 6 of them were confirmed to be inconsistent. This demonstrates that our tool can be used to find potentially erroneous CVE records and help to improve the quality of the CVE database.

5 Related Work

There have been a few efforts to apply IE to the cybersecurity domain. Most existing works focus on entity extraction for a small number of

security entities (mainly, IOCs and Vulnerabilities) from certain security text (mainly, CVEs and Tweets). Joshi et al. (Joshi et al., 2013) present a system that produces linked data from CVE records. This system can extract 6 entity types commonly found in CVEs and link the extracted instances to DBpedia entries. (Sabottke et al., 2015) proposes a Twitter-based exploit detector, which collects tweets mentioning vulnerabilities. This tool uses a simple keyword matching and monitors occurrences of the “CVE” keywords and IDs in tweets. Liao et al. (Liao et al., 2016) presents a system (iACE) for fully automated IOC extraction. iACE detects file name, IP address, and URL using regular expressions. TTPDrill (Husari et al., 2017) extracts threat actions (*i.e.*, TTP) from security reports and map them to a threat action ontology from ATT&CK and CAPEC. This tool detects threat actions from the SVO dependency structure, where the subject is a malware instance. (Yi et al., 2020) presents an NER tool for the cybersecurity domain, which is similar to our entity extraction component. They apply regular expressions, dictionary matching and a CRF classifier for about 20 different entity types and achieves about 82% F1 score.

6 Conclusion

We presented a large-scale full-stack IE system developed for the cybersecurity domain. Through careful design choices to handle the idiosyncrasies in the cybersecurity data, our system achieves high F1 scores for both entity extraction and relation extraction. We also demonstrated how our system can be used for downstream applications. Our system can help security analysts by transforming the unstructured threat reports into structured formats which can be easily consumable by subsequent security applications.

References

- Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *ACL 2016*.
- Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL 2015*.
- Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *IJCAI*.
- Robert A. Bridges, Kelly M.T. Huffer, Corinne L. Jones, Michael D. Iannacone, and John R. Goodall. 2017. Cybersecurity automated information extraction techniques: Drawbacks of current methods, and enhanced extractors. In *The 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*.
- 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*.
- Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545.
- Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992*.
- Ghaith Husari, Ehab Al-Shaer, Mohiuddin Ahmed, Bill Chu, and Xi Niu. 2017. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of CTI sources. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 103–115. ACM.
- Corinne L Jones, Robert A Bridges, Kelly MT Huffer, and John R Goodall. 2015. Towards a relation extraction framework for cyber-security concepts. In *Proceedings of the 10th Annual Cyber and Information Security Research Conference*.
- Arnav Joshi, Ravendar Lal, Tim Finin, and Anupam Joshi. 2013. Extracting cybersecurity related linked data from text. In *2013 IEEE Seventh International Conference on Semantic Computing*, pages 252–259. IEEE Computer Society.
- Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019. BERT for coreference resolution: Baselines and analysis. In *EMNLP 2019*.
- Ravendar Lal. 2013. Information Extraction of Security related entities and concepts from unstructured text. Master’s thesis, May.
- Taesung Lee and Youngja Park. 2019. Unsupervised sentence embedding using document structure-based context. In *ECML-PKDD 2019*, pages 633–647.
- Xiaojing Liao, Kan Yuan, XiaoFeng Wang, Zhou Li, Luyi Xing, and Raheem A. Beyah. 2016. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 755–766.
- Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 523–534.
- Aditya Pingle, Aritran Piplai, Sudip Mittal, Anupam Joshi, James Holt, and Richard Zak. 2019. Relext: Relation extraction using deep learning approaches for cybersecurity knowledge graph improvement. In *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*.
- Arpita Roy, Youngja Park, Taesung Lee, and Shimei Pan. 2019. Supervising unsupervised open information extraction models. In *EMNLP-IJCNLP 2019*.
- Carl Sabottke, Octavian Suciuc, and Tudor Dumitras. 2015. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 1041–1056.
- Stephen Soderland, Brendan Roof, Bo Qin, Shi Xu, Mausam, and Oren Etzioni. 2010. Adapting open information extraction to domain-specific relations. *AI Magazine*, 31(3):93–102.
- Feng Yi, Bo Jiang, Lu Wang, and Jianjun Wu. 2020. Cybersecurity named entity recognition using multi-modal ensemble learning. *IEEE Access*, 8:63214–63224.

A Appendix

A.1 Target Entity Types

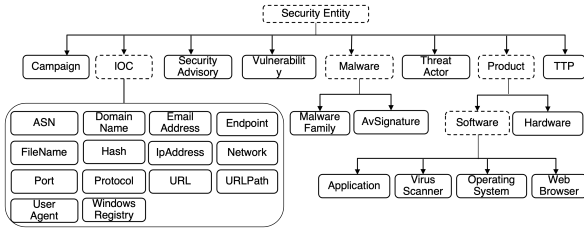


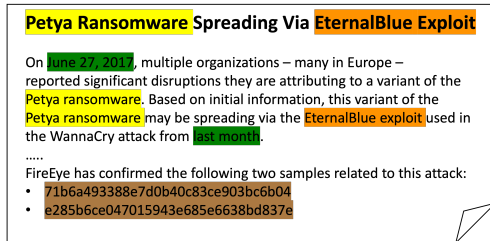
Figure 6: Target entity types and the type hierarchy in SecIE

A.2 Entity Cue Words

Entity Type	Cue Words
Campaign	breach, campaign, cyber attack, espionage, hack, scam
Malware	botnet, adware, crimeware, malware, ransomware, payload, RAT, spyware, trojan, virus, worm
ThreatActor	APT group, attacker, cyber criminal, cybercrime team, hacker, hacking group, malicious group, threat actor

Table 6: Examples of cue words for *Campaign* and *Malware*, and *ThreatActor*

A.3 Relations with Temporal Information



Arg_1	Relationship	Arg_2	Time
Petya	spread_via	EternalBlue	2017-06-27
Petya	related	WannaCry	2017-05-27
EternalBlue	used_in	WannaCry	2017-05-27
Petya	relatedHash	71b6a493388e...	2017-06-27
Petya	relatedHash	e285b6ce0470...	2017-06-27

Figure 7: Example of relation extraction with temporal information

A.4 Details of the Experimental Data

Entity Type	Count
Vulnerability	805
MalwareFamily	682
TTP	540
FileName	276
URL	239
DomainName	216
AvSignature	126
ThreatActor	105
Hash	101
SecurityAdvisory	94
Campaign	43
IpAddress	40
WindowsRegistry	10
EmailAddress	8
Endpoint	7
Network	3
Total	3,295

Table 7: Distribution of Entity Types

Relation Type	Count
Cooccurrence relation	588
OpenIE relations	306
relatedFileName	73
relatedMalware	65
relatedDomainName	45
related URL	35
relatedHash	26
relatedVulnerability	26
relatedThreatActor	11
relatedCampaign	10
relatedWindowRegistry	9
relatedCnC	6
relatedEndpoint	6
relatedIpAddress	6
relatedNetwork	2
relatedUserAgent	2
Total	1,216

Table 8: Distribution of Relation Types

A.5 Entity Extraction Performance

Entity Type	P	R	F1
AvSignature	91.9	90.5	91.2
Campaign	75.9	95.3	84.5
DomainName	91.7	91.7	91.7
EmailAddress	85.7	75.0	80.0
Endpoint	87.5	100.0	93.3
FileName	88.1	88.4	88.2
Hash	100	100	100
IpAddress	97.5	97.5	97.5
MalwareFamily	96.0	83.7	89.4
Network	100	100	100
SecurityAdvisory	96.2	54.3	69.4
ThreatActor	100	77.1	87.1
TTP	94.4	84.3	89.0
URL	97.5	98.3	97.9
Vulnerability	97.9	98.3	98.1
WindowsRegistry	100	100	100
Average	95.1	89.4	92.2

Table 9: Performance of the entity extraction models by entity types

Entity Type	BERT	SecIE
AvSignature	70.59	100.00
Campaign	66.67	100.00
DomainName	93.75	100.00
EmailAddress	0.00	66.67
Endpoint	-	-
FileName	90.20	88.00
Hash	100.00	100.00
IpAddress	100.00	100.00
MalwareFamily	62.92	91.11
Network	-	-
SecurityAdvisory	96.55	98.31
ThreatActor	22.2	93.33
TTP	67.20	65.31
URL	90.91	96.30
Vulnerability	73.42	91.82
WindowsRegistry	-	-

Table 10: Comparison of a BERT model and SecIE on 13 test documents

Deploying Unified BERT Moderation Model for E-Commerce Reviews

Ravindra Nayak N
Flipkart
Bangalore, India
ravindra.n@flipkart.com

Nikesh Garera
Flipkart
Bangalore, India
nikesh.garera@flipkart.com

Abstract

Moderation of user-generated e-commerce content has become crucial due to the large and diverse user base on the platforms. Product reviews and ratings have become an integral part of the shopping experience to build trust among users. Due to the high volume of reviews generated on a vast catalog of products, manual moderation is infeasible, making machine moderation a necessity. In this work, we described our deployed system and models for automated moderation of user-generated content. At the heart of our approach, we outline several rejection reasons for review & rating moderation and explore a unified BERT model to moderate them. We convey the importance of product vertical embeddings for the relevancy of the review for a given product and highlight the advantages of pre-training the BERT models with monolingual data to cope with the domain gap in the absence of huge labelled datasets. We observe a 4.78% F1 increase with less labelled data and a 2.57% increase in F1 score on the review data compared to the publicly available BERT-based models. Our best model In-House-BERT-vertical sends only 5.89% of total reviews to manual moderation and has been deployed in production serving live traffic for millions of users.

1 Introduction

The Internet has enabled the easy flow of information across the globe, but it has its downside too. It has led to increased hate speech and abusive communication (Veglis, 2014). It is necessary to prevent people from accessing our personal information, as it can be used for malicious purposes. The platforms that enable people to communicate and convey their opinions are also responsible for preventing profane content from affecting their users. So such platforms must have strict guidelines and strong moderation of user-generated content.

The downside of manual moderation involves inconsistency in labelling, the inability to real-time

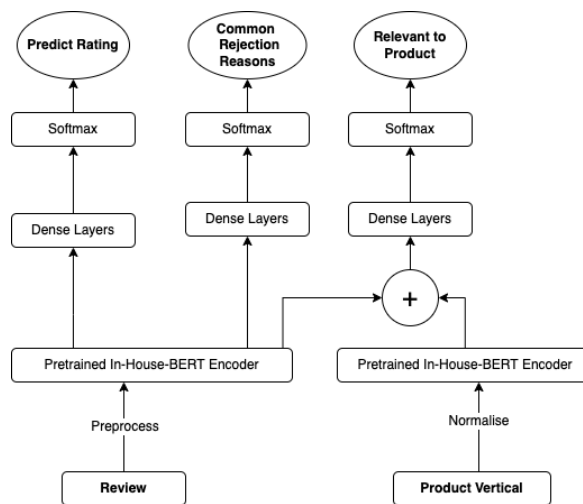


Figure 1: Model Architecture

moderation, lack of domain knowledge, and multilingual vocabulary. Due to the immense scale of the data that has been ingested on such platforms, auto-moderation becomes vital as manual moderation is not economical.

The E-commerce domain accepts multi-modal data such as text, images, and videos (Ueta et al., 2020). It is crucial to moderate them before the platform users consume the data. This paper mainly concentrates on the moderation of textual review data. Reviews and ratings build trust in the product and help platforms promote good products (Kumar, 2017). Thus eliminating reviews that do not talk about the product becomes necessary. The aim of moderating reviews is not only to detect abusive or hate speech content but also to check whether a review follows other guidelines before posting it. Before rejecting a review, it is necessary to predict the reason for rejection as feedback to the users.

We have multiple reasons for rejecting a review. These are mentioned in Table 1 along with examples. Commonly used moderation reasons include detecting profane and hate speech content (Pavlopoulos et al., 2017; Glazkova et al., 2021).

Table 1: Rejection reasons with examples

Moderation Reasons	Example 1	Example 2
approved	Just go for the good quality !! I am happy	Ok product but top coat was bad
Poorly formatted content	????!!	Nce prdddct, mast buy !!
Irrelevant review for the product	Thank you, good luck (for “watch” vertical)	I have not used it yet, dont know.. (for “mobile” vertical)
Mismatch between user-provided rating & sentiment of the review	Poor quality product (user gave a rating of 5)	Most value for money product among in ths range (user gave a rating of 1)
Profane/abusive content	Product is bull sh*t	I hate you all *****
Contains Email address(es)	abcd@gmail.co.in	Mail me raa @ outlook.com !!
Contains HTML/CSS character(s)	This is good buy.	 Link
Contains Phone Number(s)	Nine 7383 S892	Contact 92992**009 for more info
Contains URL(s)	https://docs.google.com/forms/d/e/ Please fill this form for my friend and share	https://youtu.be/uuY Unboxing video for the product

In our work, we introduce new rejection reasons (Table 1) to detect poorly formatted content, and irrelevant reviews for the product, and detect personal information like email addresses, phone numbers, and URLs. The mismatch between the rating and the sentiment of the review creates confusion in the buyer’s mind (Kumar, 2017). Hence, we predict the rating to eliminate the reviews with such a mismatch.

We start with regex parsing and list-based matching methods. These are not robust enough to capture all rejection reasons. We train a BERT (Devlin et al., 2019) based model, which predicts the rejection reasons and the rating for the given comment. We build a unified model which adheres to the review moderation guidelines set by the platform.

Publicly available base BERT (Devlin et al., 2019) is considered the baseline, and we try different architectures and configurations that help in better moderation. We use a pre-trained In-House-BERT model, which has been trained on monolingual review text and product descriptions. Pre-training helps create generic representations and adds robustness to the model (Erhan et al., 2010). We freeze embedding and initial 8 layers (Lee et al., 2019) as it helps in faster training time without degrading the model’s performance. We use product vertical / category names as an embedding to help understand the relevance of the review for that given product. We augment data with var-

ious obfuscations and noise to make the model robust to hard rejection reasons such as detecting profane/abusive content. Finally, we incorporate all these techniques to fine-tune a unified In-House-BERT moderation model to obtain an F1 score of, which is 2.57% improvement on the publicly available baseline models.

There are multiple scenarios where an auto-moderation model may fail, such as significantly morphed text, sarcastic content, or unseen data. In such a scenario, we fall back to manual moderation (Link et al., 2016). Our aim is not to fully eliminate manual moderation but instead to decrease the volume of data that goes to the moderators. When the model is not confident of its predictions, we send it for manual checks before approving it, considering it as the last line of defence.

Our major contributions from the work include:

1. Overview of our deployed text moderation system for e-commerce product reviews.
2. Unified BERT model architecture combined with deterministic approaches for moderation.
3. Demonstrating the benefits of pre-training In-House-BERT models when labelled data is scarce.
4. Illustrating the merits of adding product vertical embeddings to relevant classification heads.

- Exhibiting the importance of using hybrid approaches with the machine and manual moderation in inference setup.

2 Related work

Moderation use cases started as early as the email era and the need increased with the rise of social-media (Veglis, 2014). Traditionally hand-crafted rules were used along with basic profane word list matching. People started finding different ways to format and morph the text to bypass these systems. This paved the way sophisticated approaches with machine learning algorithms like TF-IDF (Gaydhani et al., 2018), SVM (Veloso et al., 2007) and deep learning algorithms (Saude et al., 2014; Badjatiya et al., 2017; Korencic et al., 2021; Turki and Roy, 2022).

Most of the research has been around detection of profane, hate-speech and abuse detection in the user-generated content (Pavlopoulos et al., 2017; Caselli et al., 2020; Glazkova et al., 2021). To the best of our knowledge, we haven't found any guidelines for review moderation other than detecting profane content and fake reviews (Danilchenko et al., 2022; Jindal and Liu, 2007; Rastogi and Mehrotra, 2017). We introduce sophisticated moderation guidelines for reviews and ratings in the e-commerce domain.

Dataset creation is a huge challenge as there will be imbalanced classes across various rejection reasons. Huge datasets are available for profane and hate speech content which can be curated from Twitter, Reddit, and other social media texts (Qian et al., 2019; Hee et al., 2015). These include monolingual, multilingual (i Orts, 2019; Bhattacharya et al., 2020) and code-mixed data (Bohra et al., 2018). Emojis are an important part of expressing emotions and are used to spread hate. Hate-moji (Kirk et al., 2022), is an abusive emoji dataset that has been created adversarially.

Various BERT (Devlin et al., 2019) based approaches have been taken to detect profane and hate speech content. HATE-BERT (Caselli et al., 2020), is a fine-tuned BERT model on abusive content from Reddit comments. Deep-BERT (Wadud et al., 2023), is a multilingual hate detection approach using transfer learning methods. Google has come up with their perspective 3 API (Lees et al., 2022) which uses a multilingual charformer model (Tay et al., 2021) to detect hateful content in a range of languages, domains and tasks.

Table 2: Data statistics

Dataset	Sentences	Sentences small-set
Train	34,080,768	16,384
Eval	172,544	2,234
Test	28,235	28,235

These models are generally prone to various noise attacks like adding small obfuscations or randomly changing a few characters, and its case (Hosseini et al., 2017). Significant research has been done to prevent adversarial attacks (Jain et al., 2018) on these models, and approaches like adding obfuscations and transformations to the text have shown improvements (Lees et al., 2022). Hybrid approaches of keeping humans in the loop along with the auto-moderation are also explored, which we too make use of (Link et al., 2016).

3 Proposed approach

We propose an end-to-end approach that uses a hybrid of deterministic and model-based approaches, and the data flow is shown in Figure 2.

3.1 Deterministic approaches

It is helpful to have blacklisted common profane words/phrases to do a list-based matching. We create n-gram phrases from the reviews and match them with our existing list of profane words, racial slurs, religious phrases, and political content. We maintain profane smileys, which indirectly express hate and sexual content on the platforms. We follow hybrid approaches of using the model and deterministic approaches for profane content.

We reject reviews that contain only punctuations, single letters, and random character sequences as poorly formatted content. Email addresses, phone numbers, and URLs are rejected using regex parser matching.

3.2 Domain adaptation

In the absence of abundant labelled data, we leverage the unlabelled monolingual review data by using them to pre-train the model. Pre-training helps the model understand better representation compared to the publicly available BERT. To address the domain gap, we train the BERT model from scratch as the vocabulary is updated to handle emojis and punctuations along with more relevant subwords in the e-commerce domain. We refer to this model as In-House-BERT.

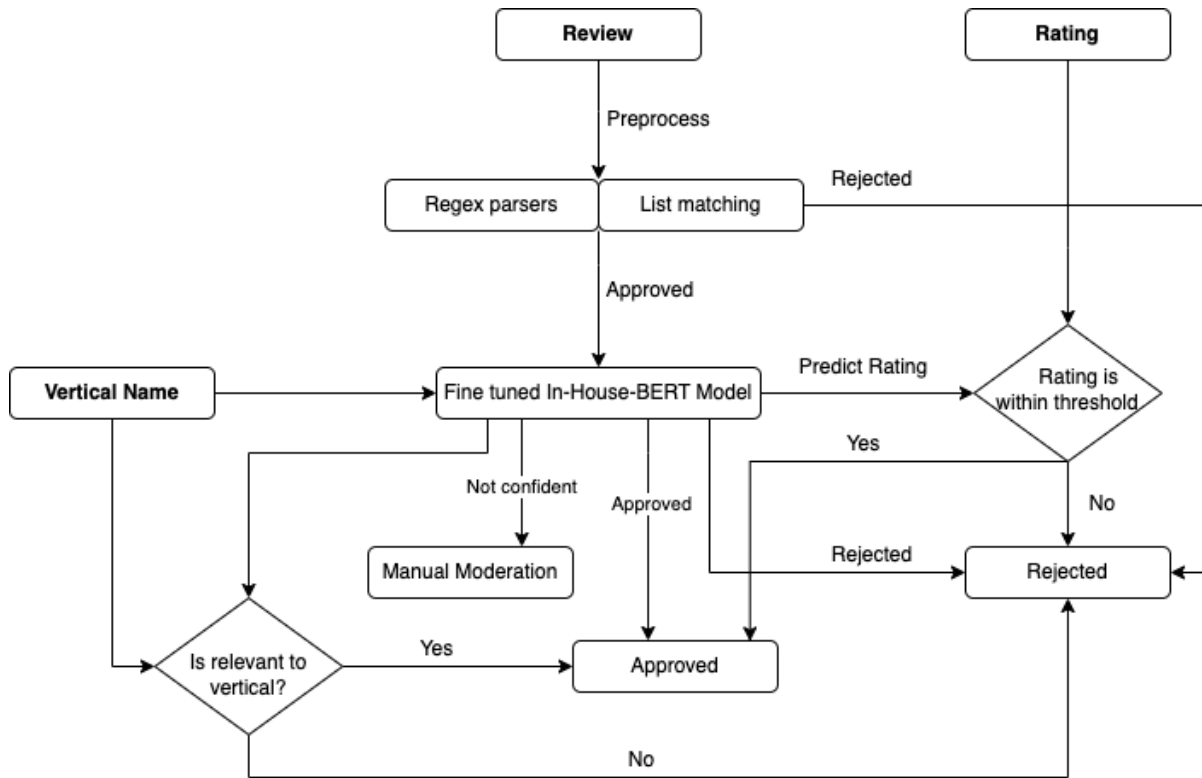


Figure 2: Dataflow of our Proposed approach

3.3 Product vertical embeddings

Product vertical information helps determine whether the given review is relevant to the product. The concatenation of review and vertical embeddings is passed to the dense layers of the classification head to detect irrelevant reviews.

3.4 Data Augmentation

Data augmentation is necessary for making the model robust to adversarial attacks. We augment only those rejection reasons which demand a high recall, i.e., profane content. We apply basic augmentations such as replacing the characters, dropping vowels, repeating characters, converting random characters to uppercase and adding profane smileys to the approved reviews. We substitute similar-looking characters such as ‘i’ with l,!, | to mimic the human perturbations.(Lees et al., 2022).

3.5 Rating prediction

Instead of having a sentiment detector separately, we reuse the rating data to predict the review’s rating. We segment the 1 to 5-star rating into 3 buckets, considering it has positive, negative, and neutral. This is a separate classification head attached to the model, which will help determine the

mismatch between the sentiment of the review and the user-given rating.

3.6 Model architecture

We develop a unified architecture, as shown in Figure 1, which can detect the various guidelines initially set to moderate the reviews. We initially have a BERT encoder(Devlin et al., 2019) which outputs a review and vertical embeddings, which are then connected to the 3 classification heads. All the heads contain dense layers followed by softmax, and they predict their respective classes. The irrelevancy detection head will get an extra vertical embedding as an input. We use the addition of 3 cross-entropy losses for back-propagation.

4 Experimental setup

4.1 Dataset

The user reviews contain text from different scripts and languages. We filter out the data to extract English text written in Roman using an in-house language classifier, which eliminates code-mixed data. We create a manually labelled corpus based on our moderation guidelines. We split the review data into train, validation, and test data, and the statistics are given in Table 2.

Table 3: F1 scores of experiments across various architectures and datasets

Models	Precision	Recall	F1 score
BERT-base	86.29	87.36	86.17
In-House-BERT	86.72	87.42	87.06
In-House-BERT-freeze	86.58	87.32	86.94
In-House-BERT-vertical	89.29	88.23	88.45
BERT-base-smallset	76.22	79.61	76.69
In-House-BERT-smallset	80.84	81.9	80.36

We create a smaller dataset of 16k training examples and name it as smallset. This dataset is created to evaluate the benefits of pre-training on monolingual data when there is a scarcity of labelled datasets. We use the same test set as before to evaluate the models.

4.2 Preprocessing

We start with the basic preprocessing of cleaning non-Roman characters and retaining emojis and punctuations. Emojis and punctuations play a vital role in understanding the review’s sentiment. We normalize the numbers to a specific format \$n\$ and \$nd\$ for ordinal numbers to help models learn generic patterns. We did an empirical analysis and found that nearly 23% of the reviews contain spelling mistakes, formatting issues, and repeating characters. Even though variations of the data will make the model robust, noise-like repetitive characters/emojis/punctuations don’t add much value to the model; hence we remove them.

4.3 Baseline and evaluation metrics

We use publicly available bert-base-cased¹ as our baseline model for evaluation with 2 classification heads, one for predicting the rating and another for the rejection reasons. This model takes a vertical name, and the text as the input and deterministic approaches are made part of the model. Training loss is the sum of cross-entropy across individual classification heads. We evaluate the models with weighted F1 scores across all the rejection reasons. The model aims to have high rejection recall while having high approval precision and decrease the volume for manual moderation, calculated by the percentage of data sent to the manual approval.

4.4 Pre-training on Monolingual data

We use product descriptions and reviews of monolingual data consisting of nearly 1B tokens to pre-

train an In-House-BERT language model with 15% masking probability and Next Sentence Prediction task. We trained the model with a learning rate of 1e-5 for 2 epochs and observed the loss converge.

4.5 Fine-tuning on labelled data

We fine-tuned the In-House-BERT model by adding 2 classification heads and trained for 2 epochs with a batch size of 512 and a learning rate of 3e-5. We tried 2 different approaches, training the whole network and freezing the embeddings and initial 4 layers. As there was no significant degradation in accuracy by freezing the weights, we used this approach for further experiments as it helped in reducing training time.

As vertical information is not necessary for common rejection reasons, we added one more classification head for detecting irrelevant product reviews by concatenating reviews and vertical embeddings before passing them to the dense layers. Finally, we train a unified model with the learning from different approaches to fine-tune a unified In-House-BERT-vertical model with 3 classification heads and freeze the initial few layers.

We experiment with a smaller test set to evaluate the importance of pre-training BERT with limited labelled data. We use similar model configurations but train on nearly 16k training samples.

4.6 Thresholds for inference setup

For inference, we set thresholds for different rejection reasons. It is always better to have lesser thresholds for stricter rejection reasons, where compromising on recall is not an option. So we empirically set the thresholds on our evaluation set and then use the same thresholds across all the models. If the model is not confident in surpassing the threshold, it will be sent to manual moderation.

¹<https://huggingface.co/bert-base-cased>

Table 4: F1 scores of experiments considering it as a binary classification problem along with Inference setup F1 scores using thresholds for better precision, and the percentage of data sent to manual moderation (lesser the better)

Models	F1 score (binary)	F1 score (with threshold)	Manual %ge
BERT-base	91.90	89.38	6.21
In-House-BERT	92.47	89.94	5.67
In-House-BERT-vertical	93.02	90.32	5.89
BERT-base-smallset	87.93	83.34	12.1
In-House-BERT-smallset	89.37	85.49	9.02

5 Results & Discussions

In Table 3, we can observe that the domain gap is being addressed using the pre-trained In-House-BERT model on smaller labelled datasets, observing an uplift of 4.78% in the F1 score. However, we don't see any significant difference with pre-training when abundant training data is available. Freezing the initial few layers of the BERT model doesn't degrade its accuracy numbers, and this can be used to reduce the training time by almost 40%. Product vertical embeddings play a better role in improving the rejection reason F1 score of individual reasons. Overall, our best model, In-House-BERT-vertical can beat the publicly available dataset by 2.57%.

5.1 Evaluating as a 2 class problem

We observe a lot of confusion for the model between rejection reasons, such as poorly formatted content being confused with irrelevant content. Further analysis revealed minor issues in the manual tagging of rejection reasons. We evaluate the model considering it as a binary classification problem with approved and reject labels. The results can be found in the first column of Table 4, where we see our best model has an F1 score of 93.02.

5.2 Impact of thresholding

It is always better to have a hybrid approach during inference because we can send the reviews for manual moderation when the model is not confident. Due to cost concerns and a longer turnaround time, it is desirable to minimise the volume of data sent to them. We set thresholds for different rejection reasons, and we observe that pre-training helps the model to be more confident at predicting the outputs reducing manual moderation load.

5.3 Deployment and Business Impact

The previously deployed system included rule-based methods and fasttext models but did not

cover all the rejection reasons we introduced. Our current deployed system also significantly reduced the volume of manually moderated reviews from 23% to 5.89%. We have tested the system up to 10 queries per second with a P95 latency of 120 ms on 2 core CPUs with 2 GB RAM. We run multiple replicas to handle the volume of live review traffic.

We measure business impact based on cost reduction and revenue generation. Reducing the manual moderation percentage led to saving millions of dollars so far and we have also externalised moderation APIs to our group companies for providing additional revenues to the company.

6 Conclusion

Pre-training BERT on large monolingual data from a similar distribution as fine-tuning gives similar results if we have large enough training data. When labelled data is scarce, we observe the advantages of pre-training the BERT models with the monolingual corpora giving a 4.78% increase in F1. Freezing the embedding layer and a few of the initial layers of the In-House-BERT model helps reduce the training time while not compromising the model's performance. Decoupling some of the rejection reasons by adding extra embeddings boosts the F1 scores. Our hybrid approach achieves an F1 score of 88.45 and sends 5.89% for manual moderation.

Limitations and Future work

As our platform supports multilingual user-generated content, it becomes essential to support multilingual, multi-script, and code-mixed moderation. We are working on the explainability of the model to convey the reasons for rejection and make the model robust to various adversarial attacks and noisy label tagging. We plan to create more data for imbalanced datasets and focus on adding other rejection reasons like sarcasm and opinion spam detection.

Acknowledgements

We thank Sudhanshu Shekhar Singh, Shreyas Shetty and Raviraj Joshi for their helpful insights and suggestions on this work. We thank Sonal Bansal, Sakshi Bhatia, Subodh Kumar, Anupam Singh, Himanshu Agarwal, Prasad Pai, Vinay Lodha, and Flipkart UGC Ops team for their constant support. We thank Amey Patil for providing the pre-trained In-House-BERT models.

References

- Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. [Deep learning for hate speech detection in tweets](#). *CoRR*, abs/1706.00188.
- Shiladitya Bhattacharya, Siddharth Singh, Ritesh Kumar, Akanksha Bansal, Akash Bhagat, Yogesh Dawer, Bornini Lahiri, and Atul Kr. Ojha. 2020. [Developing a multilingual annotated corpus of misogyny and aggression](#). *CoRR*, abs/2003.07428.
- Aditya Bohra, Deepanshu Vijay, Vinay Singh, Syed Sarfaraz Akhtar, and Manish Shrivastava. 2018. [A dataset of hindi-english code-mixed social media text for hate speech detection](#). In *Proceedings of the Second Workshop on Computational Modeling of People’s Opinions, Personality, and Emotions in Social Media, PEOPLES@NAACL-HTL 2018, New Orleans, Louisiana, USA, June 6, 2018*, pages 36–41. Association for Computational Linguistics.
- Tommaso Caselli, Valerio Basile, Jelena Mitrovic, and Michael Granitzer. 2020. [Hatebert: Retraining BERT for abusive language detection in english](#). *CoRR*, abs/2010.12472.
- Kiril Danilchenko, Michael Segal, and Dan Vilenchik. 2022. [Opinion spam detection: A new approach using machine learning and network-based algorithms](#). In *Proceedings of the Sixteenth International AAAI Conference on Web and Social Media, ICWSM 2022, Atlanta, Georgia, USA, June 6-9, 2022*, pages 125–134. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.
- Dumitru Erhan, Aaron C. Courville, Yoshua Bengio, and Pascal Vincent. 2010. [Why does unsupervised pre-training help deep learning?](#) In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 201–208. JMLR.org.
- Aditya Gaydhani, Vikrant Doma, Shrikant Kendre, and Laxmi Bhagwat. 2018. [Detecting hate speech and offensive language on twitter using machine learning: An n-gram and TFIDF based approach](#). *CoRR*, abs/1809.08651.
- Anna Glazkova, Michael Kadantsev, and Maksim Glazkov. 2021. [Fine-tuning of pre-trained transformers for hate, offensive, and profane content detection in english and marathi](#). *CoRR*, abs/2110.12687.
- Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. [Detection and fine-grained classification of cyberbullying events](#). In *Recent Advances in Natural Language Processing, RANLP 2015, 7-9 September, 2015, Hissar, Bulgaria*, pages 672–680. RANLP 2015 Organising Committee / ACL.
- Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. [Deceiving google’s perspective API built for detecting toxic comments](#). *CoRR*, abs/1702.08138.
- 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, SemEval@NAACL-HLT 2019, Minneapolis, MN, USA, June 6-7, 2019*, pages 460–463. Association for Computational Linguistics.
- 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 *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1136–1141.
- Nitin Jindal and Bing Liu. 2007. [Review spam detection](#). In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 1189–1190. ACM.
- Hannah Kirk, Bertie Vidgen, Paul Röttger, Tristan Thrush, and Scott A. Hale. 2022. [Hatemoji: A test suite and adversarially-generated dataset for benchmarking and detecting emoji-based hate](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1352–1368. Association for Computational Linguistics.
- Damir Korencic, Ipek Baris, Eugenia Fernandez, Katarina Leuschel, and Eva Salido. 2021. [To block or not to block: Experiments with machine learning for news comment moderation](#). In *Proceedings of the*

- EACL Hackashop on News Media Content Analysis and Automated Report Generation, EACL 2021, Online, April 19, 2021*, pages 127–133. Association for Computational Linguistics.
- Shashank Kumar. 2017. Research on product review analysis and spam review detection.
- Jaejun Lee, Raphael Tang, and Jimmy Lin. 2019. [What would elsa do? freezing layers during transformer fine-tuning](#). *CoRR*, abs/1911.03090.
- Alyssa Lees, Vinh Q. Tran, Yi Tay, Jeffrey Sorensen, Jai Prakash Gupta, Donald Metzler, and Lucy Vasserman. 2022. [A new generation of perspective API: efficient multilingual character-level transformers](#). *CoRR*, abs/2202.11176.
- Daniel Link, Bernd Hellingrath, and Jie Ling. 2016. [A human-is-the-loop approach for semi-automated content moderation](#). In *13th Proceedings of the International Conference on Information Systems for Crisis Response and Management, Rio de Janeiro, Brasil, May 22-25, 2016*. ISCRAM Association.
- John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. [Deep learning for user comment moderation](#). In *Proceedings of the First Workshop on Abusive Language Online, ALW@ACL 2017, Vancouver, BC, Canada, August 4, 2017*, pages 25–35. Association for Computational Linguistics.
- Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth M. Belding, and William Yang Wang. 2019. [A benchmark dataset for learning to intervene in online hate speech](#). 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 4754–4763. Association for Computational Linguistics.
- Ajay Rastogi and Monica Mehrotra. 2017. [Opinion spam detection in online reviews](#). *J. Inf. Knowl. Manag.*, 16(4):1750036:1–1750036:38.
- Marcos Rodrigues Saude, Marcelo de Medeiros Soares, Henrique Gomes Basoni, Patrick Marques Ciarelli, and Elias Oliveira. 2014. [A strategy for automatic moderation of a large data set of users comments](#). In *XL Latin American Computing Conference, CLEI 2014, Montevideo, Uruguay, September 15-19, 2014*, pages 1–7. IEEE.
- Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Prakash Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. [Charformer: Fast character transformers via gradient-based subword tokenization](#). *CoRR*, abs/2106.12672.
- Turki Turki and Sanjiban Sekhar Roy. 2022. [Novel hate speech detection using word cloud visualization and ensemble learning coupled with count vectorizer](#). *Applied Sciences*, 12(13).
- Shunya Ueta, Suganprabu Nagaraja, and Mizuki Sango. 2020. [Auto content moderation in C2C e-commerce](#). In *2020 USENIX Conference on Operational Machine Learning, OpML 2020, July 28 - August 7, 2020*. USENIX Association.
- Andreas A. Veglis. 2014. [Moderation techniques for social media content](#). In *Social Computing and Social Media - 6th International Conference, SCSM 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings*, volume 8531 of *Lecture Notes in Computer Science*, pages 137–148. Springer.
- Adriano Veloso, Wagner Meira Jr., Tiago Alves Macambira, Dorgival O. Guedes, and Hélio Marcos Paz de Almeida. 2007. [Automatic moderation of comments in a large on-line journalistic environment](#). In *Proceedings of the First International Conference on Weblogs and Social Media, ICWSM 2007, Boulder, Colorado, USA, March 26-28, 2007*.
- Md. Anwar Hussen Wadud, Muhammad F. Mridha, Jungpil Shin, Kamruddin Nur, and Alope Kumar Saha. 2023. [Deep-bert: Transfer learning for classifying multilingual offensive texts on social media](#). *Comput. Syst. Sci. Eng.*, 44(2):1775–1791.

A Obfuscation techniques

Augmentation techniques are used to create more data for profane and hate speech content by adding multiple obfuscation techniques described in Table 5. Data augmentation certainly gives a boost to profane content F1 scores by 18%.

Table 5: Various data augmentation techniques that we used on an example profane word "bullshit"

Technique	Augmented phrases
Replace characters with *	bu**sh*t, bullsh*t
Drop vowels randomly	bllsht, bullsht
Repeating characters	bullllshiiiiit
Random case changing	buLLshIT
Add random spaces	bu llshi t, bull shit
Replace similar-looking characters	bull 5h!t, bu!! śhît

SimANS: Simple Ambiguous Negatives Sampling for Dense Text Retrieval

Kun Zhou^{1,3†}, Yeyun Gong⁴, Xiao Liu⁴, Wayne Xin Zhao^{2,3*}, Yelong Shen⁵, Anlei Dong⁵,
Jingwen Lu⁵, Rangan Majumder⁵, Ji-Rong Wen^{2,3}, Nan Duan⁴, Weizhu Chen⁵

¹School of Information, Renmin University of China,

²Gaoling School of Artificial Intelligence, Renmin University of China,

³Beijing Key Laboratory of Big Data Management and Analysis Methods,

⁴Microsoft Research, ⁵Microsoft

Abstract

Sampling proper negatives from a large document pool is vital to effectively train a dense retrieval model. However, existing negative sampling strategies suffer from the uninformative or false negative problem. In this work, we empirically show that according to the measured relevance scores, the negatives ranked around the positives are generally more informative and less likely to be false negatives. Intuitively, these negatives are not too hard (*may be false negatives*) or too easy (*uninformative*). They are the ambiguous negatives and need more attention during training. Thus, we propose a simple ambiguous negatives sampling method, SimANS, which incorporates a new sampling probability distribution to sample more ambiguous negatives. Extensive experiments on four public and one industry datasets show the effectiveness of our approach. We made the code and models publicly available in <https://github.com/microsoft/SimXNS>.

1 Introduction

Dense text retrieval, which uses low-dimensional vectors to represent queries and documents and measure their relevance, has become a popular topic (Karpukhin et al., 2020; Luan et al., 2021) for both researchers and practitioners. It can improve various downstream applications, *e.g.*, web search (Brickley et al., 2019; Qiu et al., 2022) and question answer (Izacard and Grave, 2021). A key challenge for training a dense text retrieval model is how to select appropriate negatives from a large document pool (*i.e.*, negative sampling), as most existing methods use a contrastive loss (Karpukhin et al., 2020; Xiong et al., 2021) to encourage the model to rank positive documents higher than negatives. However, the commonly-used negative sampling strategies, namely random negative sampling (Luan et al., 2021; Karpukhin et al., 2020)

(using random documents in the same batch) and top- k hard negatives sampling (Xiong et al., 2021; Zhan et al., 2021) (using an auxiliary retriever to obtain the top- k documents), have their limitations. Random negative sampling tends to select uninformative negatives that are rather easy to be distinguished from positives and fail to provide useful information (Xiong et al., 2021), while top- k hard negatives sampling may include false negatives (Qu et al., 2021), degrading the model performance.

Motivated by these problems, we propose to sample the *ambiguous negatives*¹ that are neither too easy (uninformative) nor too hard (potential false negatives). Our approach is inspired by an empirical observation from experiments (in §3) using gradients to assess the impact of data instances on deep models (Koh and Liang, 2017; Pruthi et al., 2020): according to the measured relevance scores using the dense retrieval model, negatives that rank lower are mostly uninformative, as their gradient means are close to zero; negatives that rank higher are likely to be false negatives, as their gradient variances are significantly higher than expected. Both types of negatives are detrimental to the convergence of deep matching models (Xiong et al., 2021; Qu et al., 2021). Interestingly, we find that the negatives ranked around positive examples tend to have relatively larger gradient means and smaller variances, indicating that they are informative and have a lower risk of being false negatives, thus probably being high-quality ambiguous negatives.

Based on these insights, we propose a **Simple Ambiguous Negative Sampling** method, namely **SimANS**, for improving deep text retrieval. Our main idea is to design a sampling probability distribution that can assign higher probabilities to the ambiguous negatives while lower probabilities to the

¹We call them ambiguous negatives following the definition of ambiguous examples (Swayamdipta et al., 2020; Meissner et al., 2021), referring to the instances that are neither too hard nor too easy to learn.

[†] This work was done during internship at MSRA.

* Corresponding author, email: batmanfly@gmail.com.

possible false and uninformative negatives, based on the differences of the relevance scores between positives and candidate negatives. We also incorporate two hyper-parameters to better adjust the peak and density of the sampling probability distribution. Our approach is simple and flexible, which can be easily applied to various dense retrieval models and combined with other effective techniques, *e.g.*, knowledge distillation (Qu et al., 2021) and adversarial training (Zhang et al., 2021).

To validate the effectiveness of SimANS, we conduct extensive experiments on four public datasets and one industrial dataset collected from Bing search logs. Experimental results show that SimANS can improve the performance of competitive baselines, including state-of-the-art methods.

2 Preliminary

Dense Text Retrieval. Given a query q , the dense text retrieval task aims to retrieve the most relevant top- k documents $\{d_i\}_{i=1}^k$ from a large candidate pool \mathcal{D} . To achieve it, the dual-encoder architecture is widely used due to its efficiency (Reimers and Gurevych, 2019; Karpukhin et al., 2020). It consists of a query encoder E_q and a document encoder E_d to map the query q and document d into k -dimensional dense vectors \mathbf{h}_q and \mathbf{h}_d , respectively. Then, the semantic relevance score of q and d can be computed using dot product as

$$s(q, d) = \mathbf{h}_q \cdot \mathbf{h}_d. \quad (1)$$

Recent works mostly adopt pre-trained language models (PLMs) (Devlin et al., 2019) as the two encoders, and utilize the representations of the [CLS] token as dense vectors.

Training with Negative Sampling. The training objective of dense text retrieval task is to pull the representations of the query q and relevant documents \mathcal{D}^+ together (as positives), while pushing apart irrelevant ones $\mathcal{D}^- = \mathcal{D} \setminus \mathcal{D}^+$ (as negatives). However, the irrelevant documents are from a large document pool, which would lead to millions of negatives. To reduce the unreachable training cost, negative sampling has been widely used. Previous works either randomly sample negatives (Karpukhin et al., 2020), or select the top- k hard negatives ranked by BM25 or the dense retrieval model itself (Xiong et al., 2021; Qu et al., 2021), denoted as $\tilde{\mathcal{D}}^-$. Then, the optimization ob-

jective can be formulated as:

$$\theta^* = \arg \min_{\theta} \sum_q \sum_{d^+ \in \mathcal{D}^+} \sum_{d^- \in \tilde{\mathcal{D}}^-} L(s(q, d^+), s(q, d^-)), \quad (2)$$

where $L(\cdot)$ is the loss function.

3 Motivation Study

We first analyze the uninformative and false negative problems from the perspective of gradients. Then, we perform an empirical study to test how gradients of negatives change *w.r.t.* ranks according to measured relevance scores using a dense retrieval model, and find that the gradients of negatives ranked near positives have relatively larger means and smaller variances.

3.1 Analysis for Gradients of Negatives

Existing dense retrieval methods (Karpukhin et al., 2020; Xiong et al., 2021) commonly incorporate the binary cross entropy (BCE) loss to compute gradients², where the relevance scores of a positive and sampled negatives are usually normalized by the softmax function. In this way, the gradients of model parameters θ are computed by

$$\nabla_{\theta} l(q, d) = \begin{cases} (s_n(q, d) - 1) \nabla_{\theta} s_n(q, d) & \text{if } d \in \mathcal{D}^+ \\ s_n(q, d) \nabla_{\theta} s_n(q, d) & \text{if } d \in \mathcal{D}^- \end{cases}$$

where $s_n(q, d)$ is the normalized value of $s(q, d)$ and is within $[0, 1]$. Based on it, we review the gradients of uninformative and false negatives. Uninformative negatives can be easily distinguished by dense retrieval models, and are more likely to be selected by random sampling (Xiong et al., 2021). As their normalized relevance scores are usually rather small, *i.e.*, $s_n(q, d) \rightarrow 0$, their gradient means will be bounded into near-zero values, *i.e.*, $\nabla_{\theta} l(q, d) \rightarrow 0$. Such near-zero gradients are also uninformative and contribute little to model convergence. False negatives are usually semantically similar to positives, and are more likely to be selected by top- k hard negatives sampling (Qu et al., 2021). Therefore, for the gradients of false negatives and positives, the right terms $\nabla_{\theta} s_n(q, d)$ may be similar, while the left terms are greater than zero and less than 0, respectively. As a result, the variance of gradients will be larger, which may cause the optimization of parameters to be unstable. Furthermore, existing works (Katharopoulos and Fleuret, 2018; Johnson and Guestrin, 2018) have

²In this work, we perform the analysis using BCE loss, and such analysis can also be extended to other loss functions.

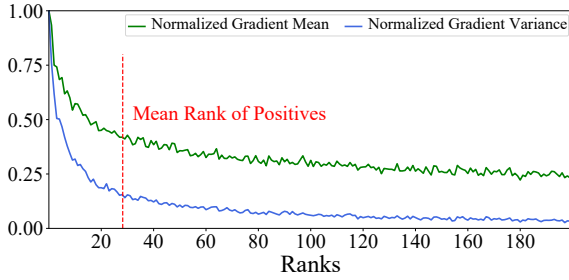


Figure 1: The mean and variance of gradients change curves *w.r.t.* the ranks of negatives on MS-MARCO Passage Ranking dataset using AR2 (Zhang et al., 2021).

theoretically proved that larger gradient variance is detrimental to model convergence.

3.2 Empirical Study on Gradients of Negatives *w.r.t.* Relevance Scores

Although we have analyzed that the harmful influence of uninformative and false negatives derives from the smaller means and larger variances of gradients respectively, it is time-consuming to compute gradients of all candidate negatives to identify and remove them. Here, we empirically study if the query-document relevance scores can be leveraged to avoid sampling these harmful negatives.

Experimental Setup. We use AR2 (Zhang et al., 2021) as the retrieval model and investigate its gradients on the development set of MS-MARCO Passage Ranking dataset (Nguyen et al., 2016). Concretely, for each query, we rank all negatives according to their relevance scores, and compute the means and variances of gradients of all negatives in the same rank³. To better show the tendency *w.r.t.* ranks of relevance scores, we normalize the means and variances of gradients by dividing the maximum values, and only report the results of top 200 ranked negatives.

Results and Findings. As shown in Figure 1, the mean and variance of gradients will gradually decrease with the increase of the rank. Despite that, the gradient means of the top 200 negatives are still in the same order of magnitude (1.0 \rightarrow 0.25), while the gradient variances of the top 10 ranked negatives are significantly larger than others. The reason is that the higher-ranking negatives have larger probabilities to be false negatives. Besides, a surprising finding is that the mean rank of posi-

³As AR2 adopts ERNIE-2.0 (Sun et al., 2020) as the backbone that has millions of parameters, we only compute gradients on the parameters of its last layer for efficiency.

tives is approximate the boundary point of the high gradient variance part and the negatives near it can produce relatively larger gradient means and lower gradient variances. It means that they are high-quality *ambiguous negatives* that can balance the informativeness and the risk of being false negatives. Therefore, it is promising to rely on the relevance scores of positives and candidate negatives to devise more effective negative sampling methods for training dense retrieval models.

4 Approach

Based on the findings in §3, we conjecture that the ambiguous negatives ranked near positives according to relevance scores are high-quality negatives, as they are neither too easy (uninformative) nor too hard (may be false negatives). Therefore, we propose a simple ambiguous negative sampling method, namely SimANS.

4.1 Ambiguous Negative Sampling

To focus on sampling ambiguous negatives, we design a new sampling probability distribution that can estimate the influence of each negative using the dense retrieval models. As follows, we first devise a general sampling distribution and then propose its simple and efficient implementation.

General Sampling Distribution. We draw the following conclusions from our results about how to choose a good sampling probability distribution for negatives: (1) Negatives that are clearly irrelevant and have low relevance scores should be sampled less frequently; (2) Negatives that are highly relevant and have high relevance scores should also be sampled less frequently, because they are more likely to be positives in disguise; (3) Negatives that are uncertain and have relevance scores similar to positives should be sampled more frequently, because they provide useful information and have a lower chance of being false negatives. We propose a general formula for negative sampling probability that reflects these principles:

$$p_i \propto f(|s(q, d_i) - \bar{s}(q, d^+) - b|), \forall d_i \in \mathcal{D} \setminus \mathcal{D}^+, \quad (3)$$

where $f(\cdot)$ is a function to determine the tendency of the probability distribution, b is a hyper-parameter to control the peak of the distribution, $\bar{s}(q, d^+)$ is the mean relevance score of all positives with the query. $f(\cdot)$ should be a monotone

decreasing function (e.g., e^{-x}). In this way, the negatives with the relevance scores close to positives can be assigned with larger probabilities, while others with smaller or larger scores will be punished with smaller probabilities. Such a distribution can satisfy the required three characteristics.

Simple Negative Sampling Distribution. We rely on several empirical priors to determine a simple and efficient implementation of the above sampling probability distribution. Generally, the relevance scores of positives and negatives are bounded by the modulus of dense vectors, hence they are mostly in a same order of magnitude. To ensure that the probabilities of ambiguous negatives should be significantly larger than other ones, we choose the exponential function to implement $f(\cdot)$. As a large proportion of negatives from $\mathcal{D} \setminus \mathcal{D}^+$ are uninformative ones, their smaller relevance scores would lead to near-zero probabilities using the exponential function. Therefore, we can reduce the computation cost by narrowing the negative candidates into the top- k ranked negatives $\tilde{\mathcal{D}}^-$. In addition, to further reduce the cost, we also replace the mean relevance score of all positives $\bar{s}(q, \mathcal{D}^+)$ by the score of a randomly sampled positive $s(q, \tilde{d}^+)$. Finally, we can reformulate the sampling probability distribution in equation (3) as:

$$p_i \propto \exp(-a(s(q, d_i) - s(q, \tilde{d}^+) - b)^2), \forall d_i \in \tilde{\mathcal{D}}^-, \quad (4)$$

where a is a hyper-parameter to control the density of the distribution, $\tilde{d}^+ \in \mathcal{D}^+$ is a randomly sampled positive, $\tilde{\mathcal{D}}^-$ is the top- k ranked negatives. In this way, the complexity of computing the sampling probability distribution will be reduced into $O(k)$, where $k \ll |\mathcal{D}|$ and we set it to 100.

4.2 Overview and Discussion

Overview. Given a mini-batch, SimANS contains three major steps to obtain the ambiguous negatives. The first step is the same as previous top- k hard negatives sampling methods (Xiong et al., 2021; Qu et al., 2021) that select the top- k ranked negatives $\tilde{\mathcal{D}}^-$ from the candidate pool $\mathcal{D} \setminus \mathcal{D}^+$ using an ANN search tool (e.g., FAISS (Johnson et al., 2019)). Second, we compute the sampling probabilities for all the top- k negatives using equation (4). To reduce the time cost, we can pre-compute them in the first step. Finally, we sample the ambiguous negatives *w.r.t.* their sampling probabilities. We present the overall algorithm in Algorithm 1.

Algorithm 1: The algorithm of SimANS.

Input: Queries and their positive documents $\{(q, \mathcal{D}^+)\}$, document pool \mathcal{D} , pre-learned dense retrieval model M

- 1 Build the ANN index on \mathcal{D} using M .
- 2 Retrieve the top- k ranked negatives $\tilde{\mathcal{D}}^-$ for each query with their relevance scores $\{s(q, d_i)\}$ from \mathcal{D} .
- 3 Compute the relevance scores of each query and its positive documents $\{s(q, \mathcal{D}^+)\}$.
- 4 Generate the sampling probabilities of retrieved top- k negatives $\{p_i\}$ for each query using Eq. 3.
- 5 Construct new training data $\{(q, \mathcal{D}^+, \tilde{\mathcal{D}}^-)\}$.
- 6 **while** M has not converged **do**
- 7 Sample a batch from $\{(q, \mathcal{D}^+, \tilde{\mathcal{D}}^-)\}$.
- 8 Sample ambiguous negatives for each instance from the batch according to $\{p_i\}$.
- 9 Optimize parameters of M using the batch and sampled negatives.
- 10 **end**

Note that our proposed SimANS is a negative sampling method and applicable to a variety of dense retrieval methods.

Relationship with Other Methods. SimANS aims to sample the ambiguous negatives that rank close to the positives according to relevance scores for improving the training of dense retrieval models. It is a general framework that several previous negative sampling methods can be included:

- **Choosing negative examples randomly** means picking them from a big collection of documents with equal chances for each one. We can also use our method to do this by setting $b = s(q, d_i) - s(q, \tilde{d}^+)$ and making $\tilde{\mathcal{D}}^-$ include all the documents in the collection. But this is not a good idea, because most of the documents in the collection are not relevant to the query and do not help us learn from the feedback. They are easy to sample but not useful for training.

- **Top- k hard negatives sampling** utilizes an auxiliary retriever (e.g., BM25 (Karpukhin et al., 2020) or DPR (Xiong et al., 2021)) to rank all negative candidates and pick the top- k ones as negatives. By setting $b = -s(q, \tilde{d}^+)$ and $a = -\inf$, our method can also produce extremely large probabilities to the top- k negatives. Whereas, the top- k ones have a higher risk to be false negatives, which are harmful to convergence.

5 Experiments

5.1 Experimental Setting

We extensively evaluate SimANS by conducting experiments on three public passage retrieval datasets:

Datasets	Training	Dev	Test	Documents
NQ	58,880	8,757	3,610	21,015,324
TQ	60,413	8,837	11,313	21,015,324
MS Pas	502,939	6,980	-	8,841,823
MS Doc	367,013	5,193	-	3,213,835
Bing	1,861,102	8,013	-	5,335,927

Table 1: Statistics of the five text retrieval datasets.

Natural Question (NQ) (Kwiatkowski et al., 2019), Trivia QA (TQ) (Joshi et al., 2017) and MS-MARCO Passage Ranking (MS Pas) (Nguyen et al., 2016), a public document retrieval dataset: MS-MARCO Document Ranking (MS Doc) (Nguyen et al., 2016), and an industry dataset that is collected from Bing search logs. Their statistics are shown in Table 1. The details of datasets, baselines and implementations are presented in Appendix.

5.2 Results Analysis

Performance on Public Retrieval Datasets. Table 2 and Table 3 show the experimental results on three public passage retrieval datasets. First, we can see that AR2 outperforms most baseline methods on all datasets. AR2 incorporates an adversarial training framework to iteratively improve the retriever and ranker. Second, SimANS can further improve the performance of AR2, and outperform all baselines in terms of all the metrics across all datasets. SimANS only incorporates a new negative sampling strategy based on AR2, which aims to sample the ambiguous negatives that are neither too hard (potential false negatives) or too easy (uninformative). According to the findings in §3, such a way can alleviate the uninformative and false negative problems that are frequently encountered in commonly-used random and top- k negatives sampling methods, and is able to sample high-quality negatives that contribute more to the model convergence. Besides, the improvements of SimANS on AR2 are larger in MS Pas and Doc datasets than others. The reason is that the two datasets are collected from real-world search logs that suffer severely from the false negative problem, whereas SimANS is capable of alleviating this problem and provides better negatives for training.

Performance on Bing Industry Dataset. For the Bing industry dataset, we adopt a dual-encoder mBERT (Devlin et al., 2019) as the baseline model to deal with multilingual queries and documents, and implement different negative sampling strate-

gies on it. We simply evaluate the last checkpoint after training and report the results on the development set. As shown in Table 4, after applying the top- k hard negatives sampling, the performance of the baseline model is improved by a large margin. It indicates that hard negatives are more effective than randomly sampled ones. Furthermore, we can see that SimANS outperforms all other negative sampling methods, especially in Hit@5 (2% absolute improvement). It demonstrates the effectiveness of SimANS in industrial scenarios. As a comparison, SimANS is able to alleviate the uninformative and false negatives problems that the random and top- k negatives sampling strategies may suffer, respectively.

5.3 Further Analysis

Applying SimANS to Other Models. Since SimANS is a general negative sampling strategy, it can be applied to a variety of dense retrieval methods. Thus, in this part, we implement SimANS on two representative methods, ANCE (Xiong et al., 2021) and RocketQA (Qu et al., 2021), as they adopt effective techniques as asynchronous index refresh and knowledge distillation, respectively. We only replace the negative sampling strategies in these methods with SimANS and conduct experiments on TQ and NQ datasets. As shown in Table 5, our approach can consistently improve the performance of the two methods. It shows that SimANS is general to various dense retrieval methods with different techniques and can provide more high-quality negatives to improve their performance.

Variation Study. Our proposed SimANS incorporates a new negative sampling probability distribution that is based on the differences between the query-document relevance scores of positives and negative candidates. To verify the effectiveness of this distribution, we design two variations of SimANS: (1) **Doc-Sim** that leverages the document-document relevance scores between positives and negative candidates to replace the query-document relevance scores; (2) **Nearest-K** that directly picks the top- k nearest negatives according to the differences of query-document relevance scores instead of sampling. We implement these variations on AR2 and conduct experiments on the development set of MS Pas dataset. As shown in Table 6, SimANS outperforms all these variations. It indicates the effectiveness of our devised ambigu-

Method	NQ			TQ			MS Pas		
	R@5	R@20	R@100	R@5	R@20	R@100	MRR@10	R@50	R@1k
BM25 (Yang et al., 2017)	-	59.1	73.7	-	66.9	76.7	18.7	59.2	85.7
GAR (Mao et al., 2021)	60.9	74.4	85.3	73.1	80.4	85.7	-	-	-
doc2query (Nogueira et al., 2019b)	-	-	-	-	-	-	21.5	64.4	89.1
DeepCT (Dai and Callan, 2019)	-	-	-	-	-	-	24.3	69.0	91.0
docTTTTTquery (Nogueira et al., 2019a)	-	-	-	-	-	-	27.7	75.6	94.7
DPR (Karpukhin et al., 2020)	-	78.4	85.3	-	79.3	84.9	-	-	-
ANCE (Xiong et al., 2021)	71.8	81.9	87.5	-	80.3	85.3	33.0	81.1	95.9
COIL (Gao et al., 2021a)	-	-	-	-	-	-	35.5	-	96.3
ME-BERT (Luan et al., 2021)	-	-	-	-	-	-	33.8	-	-
Joint top- k (Sachan et al., 2021)	72.1	81.8	87.8	74.1	81.3	86.3	-	-	-
Individual top- k (Sachan et al., 2021)	75.0	84.0	89.2	76.8	83.1	87.0	-	-	-
RocketQA (Qu et al., 2021)	74.0	82.7	88.5	-	-	-	37.0	85.5	97.9
RDR (Yang and Seo, 2020)	-	82.8	88.2	-	82.5	87.3	-	-	-
RocketQAv2 (Ren et al., 2021b)	75.1	83.7	89.0	-	-	-	38.8	86.2	98.1
PAIR (Ren et al., 2021a)	74.9	83.5	89.1	-	-	-	37.9	86.4	98.2
DPR-PAQ (Oğuz et al., 2022)	74.2	84.0	89.2	-	-	-	31.1	-	-
Condenser (Gao and Callan, 2021)	-	83.2	88.4	-	81.9	86.2	36.6	-	97.4
coCondenser (Gao and Callan, 2022)	75.8	84.3	89.0	76.8	83.2	87.3	38.2	-	98.4
ERNIE-Search (Lu et al., 2022)	77.0	85.3	89.7	-	-	-	40.1	87.7	98.2
AR2 (Zhang et al., 2021)	<u>77.9</u>	<u>86.0</u>	<u>90.1</u>	<u>78.2</u>	<u>84.4</u>	<u>87.9</u>	39.5	<u>87.8</u>	<u>98.6</u>
AR2+SimANS	78.6	86.2	90.3	78.6	84.6	88.1	40.9	88.7	98.7

Table 2: Performance on the test sets of NQ and TQ, and the development set of MS Pas. The results of baselines are from original papers. The best and second-best methods are marked in bold and underlined, respectively.

Method	MRR@10	R@100
BM25	0.279	0.807
DPR (Karpukhin et al., 2020)	0.320	0.864
ANCE (Xiong et al., 2021)	0.377	0.894
STAR (Zhan et al., 2021)	0.390	0.913
ADORE (Zhan et al., 2021)	0.405	<u>0.919</u>
AR2 (Zhang et al., 2021)	<u>0.418</u>	0.914
AR2+SimANS	0.431	0.923

Table 3: Performance on MS Doc development set.

Method	R@5	R@20	R@100
Baseline+Random Neg	39.5	59.0	76.2
Baseline+top- k Neg	57.1	73.5	85.1
Baseline+SimANS	59.1	74.9	85.6

Table 4: Experimental results on Bing Industry dataset.

ous negative sampling probability distribution. For Doc-Sim, it is likely to select the false negatives that have similar semantics to positives, hurting the model performance. For Nearest-K, as it always selects fixed negatives, it may cause overfitting.

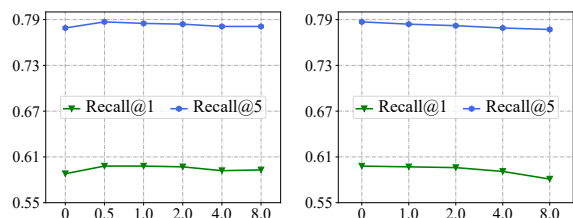
Parameter Tuning. Our SimANS has two important hyper-parameters to tune, a and b , which control the density and peak of the sampling probability distribution, respectively. Here, we investigate the performance change of SimANS on AR2 *w.r.t.* different a and b on NQ dataset. As shown in Figure 2, our approach achieves the best performance when $a = 0.5$ and $b = 0$. It indicates that

Method	TQ		NQ	
	R@5	R@20	R@5	R@20
ANCE	72.4	80.3	71.8	81.9
ANCE+SimANS	74.8	82.1	74.3	83.0
RocketQA	76.1	83.0	74.0	82.7
RocketQA+SimANS	77.1	83.6	76.7	84.8

Table 5: The retrieval performance of applying our method on other baselines on TQ and NQ datasets

Method	MRR@10	R@1	R@50	R@1k
AR2	39.5	26.4	87.8	98.6
AR2+Doc-Sim	40.1	27.3	88.0	98.6
AR2+Nearest-K	40.5	27.6	88.5	98.7
AR2+SimANS	40.9	28.2	88.7	98.7

Table 6: The variation study of our method in AR2 on MS Pas development set.



(a) a : density hyper-parameter (b) b : peak hyper-parameter

Figure 2: Performance comparison *w.r.t.* hyper-parameters a and b on NQ dataset.

when the maximum point of the distribution has the same relevance score as the positive, the nega-

Ratio	AR2		AR2+SimANS	
	R@5	Latency	R@5	Latency
1 : 1	76.4	210ms	77.5	210ms
1 : 5	76.9	330ms	78.1	340ms
1 : 11	77.1	510ms	78.3	540ms
1 : 15	77.9	630ms	78.7	650ms

Table 7: The retrieval performance and training latency *w.r.t.* different sampled negative ratios on NQ dataset.

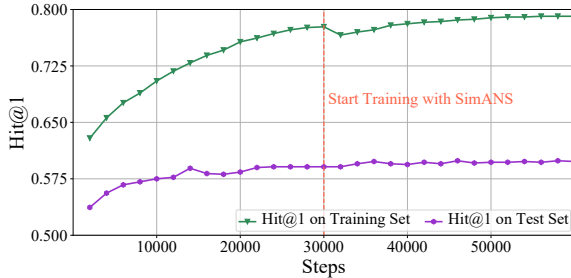


Figure 3: Hit@1 of AR2+SimANS on training and test sets of NQ *w.r.t.* training steps.

tive sampling probability distribution can produce more high-quality negatives. Moreover, we notice that the model performance is not very sensitive to the two hyper-parameters if they are properly set within a certain range.

Impact of the Sampled Negative Ratio. We investigate the impact of the sampled negative ratio $1 : k$ on retrieval performance and training latency per batch of SimANS on AR2. As shown in Table 7, with the increase of the sampled negative number, the performance improves consistently while the training latency increases. Besides, SimANS just slightly increases the training latency of AR2. It is because we can pre-compute the sampling probabilities before training, which avoids time-consuming computation during training.

Performance *w.r.t.* Training Steps. Our approach requires continually training the model parameters that have been pre-trained by the original dense retrieval method. Here, we investigate the performance changes of the dense retrieval method before and after using SimANS *w.r.t.* the training steps. We conduct experiments on AR2 and show the Hit@1 metric on NQ dataset in Figure 3. First, we can see that with the increase of the training steps, the performance of AR2 on training and test sets improves simultaneously. After applying our SimANS, we can see that the performance further improves, especially in the training set (0.777 \rightarrow 0.791). It indicates that our ap-

proach is capable of improving the fitting of the training set, and such an improvement can also generalize to the test set.

6 Conclusion

We investigated how the gradient statistics of negative documents affect their relevance ranking for dense text retrieval. We discovered that negative documents with high gradient means and low gradient variances are more likely to be ambiguous negatives, which are informative and less prone to false negatives. Based on this insight, we proposed SimANS, a novel negative sampling method that balances the difficulty of negative examples by adjusting their sampling probabilities. SimANS improved the performance of various dense retrieval models on four public and one industrial datasets. We plan to apply our method to other information retrieval tasks, such as personal recommendation, and to develop better pre-training schemes for dense text retrieval in the future.

Acknowledgement

Kun Zhou, Wayne Xin Zhao and Ji-Rong Wen are supported by Beijing Natural Science Foundation under Grant No. 4222027, and National Natural Science Foundation of China under Grant No. 61872369, Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098, and the Outstanding Innovative Talents Cultivation Funded Programs 2021. Xin Zhao is the corresponding author.

Ethical Consideration

In this section, we discuss the ethical considerations of this work from the following two aspects. First, for intellectual property protection, the code, data and dense retrieval models adopted from previous works are granted for research-purpose usage. Second, since PLMs have been shown to capture certain biases from the pre-trained corpus (Bender et al., 2021), there is a potential problem about biases that are from the use of PLMs in our approach. There are increasing efforts to address this problem in the community (Ross et al., 2021).

References

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models

- be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.
- Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *WWW*.
- Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for ir with contextual neural language modeling. In *SIGIR*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.
- Luyu Gao and Jamie Callan. 2021. Is your language model ready for dense representation fine-tuning? *arXiv preprint arXiv:2104.08253*.
- Luyu Gao and Jamie Callan. 2022. Unsupervised corpus aware language model pre-training for dense passage retrieval. In *ACL*, pages 2843–2853.
- Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021a. COIL: revisit exact lexical match in information retrieval with contextualized inverted list. In *NAACL-HLT*.
- Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. 2021b. Scaling deep contrastive learning batch size under memory limited setup. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepLANLP-2021)*.
- Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. 2021. Efficiently teaching an effective dense retriever with balanced topic aware sampling. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–122.
- Wu Hong, Zhuosheng Zhang, Jinyuan Wang, and Hai Zhao. 2022. Sentence-aware contrastive learning for open-domain passage retrieval. In *ACL*, pages 1062–1074.
- Gautier Izacard and Édouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.
- Tyler B Johnson and Carlos Guestrin. 2018. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31.
- Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP*.
- Angelos Katharopoulos and François Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR.
- Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. 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. *Trans. Assoc. Comput. Linguistics*, 7:452–466.
- Yuxiang Lu, Yiding Liu, Jiayang Liu, Yunsheng Shi, Zhengjie Huang, Shikun Feng Yu Sun, Hao Tian, Hua Wu, Shuaiqiang Wang, Dawei Yin, et al. 2022. Ernie-search: Bridging cross-encoder with dual-encoder via self on-the-fly distillation for dense passage retrieval. *arXiv preprint arXiv:2205.09153*.
- Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2021. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345.
- Kelong Mao, Zhicheng Dou, and Hongjin Qian. 2022. Curriculum contrastive context denoising for few-shot conversational dense retrieval. In *SIGIR*, pages 176–186.
- Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2021. Generation-augmented retrieval for open-domain question answering. In *ACL*.
- Johannes Mario Meissner, Napat Thumwanit, Saku Sugawara, and Akiko Aizawa. 2021. Embracing ambiguity: Shifting the training target of nli models. In *ACL*, pages 862–869.
- Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *EMNLP*, pages 5783–5797.
- 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 *CoCo@ NIPS*.

- Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. 2019a. From doc2query to docttttquery. *Online preprint*.
- Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv preprint arXiv:1904.08375*.
- Barlas Oğuz, Kushal Lakhotia, Anchit Gupta, Patrick Lewis, Vladimir Karpukhin, Aleksandra Piktus, Xilun Chen, Sebastian Riedel, Wen-tau Yih, Sonal Gupta, et al. 2022. Domain-matched pre-training tasks for dense retrieval. In *Findings of NAACL*.
- Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930.
- Yifu Qiu, Hongyu Li, Yingqi Qu, Ying Chen, Qiaoqiao She, Jing Liu, Hua Wu, and Haifeng Wang. 2022. Dureader_retrieval: A large-scale chinese benchmark for passage retrieval from web search engine. *arXiv preprint arXiv:2203.10232*.
- Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *NAACL-HLT*.
- Ori Ram, Gal Shachaf, Omer Levy, Jonathan Berant, and Amir Globerson. 2022. Learning to retrieve passages without supervision. In *NAACL*.
- 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.
- Ruiyang Ren, Shangwen Lv, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021a. PAIR: leveraging passage-centric similarity relation for improving dense passage retrieval. In *Findings of ACL/IJCNLP*.
- Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021b. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. In *EMNLP*, pages 2825–2835.
- Candace Ross, Boris Katz, and Andrei Barbu. 2021. Measuring social biases in grounded vision and language embeddings. In *NAACL*, pages 998–1008.
- Devendra Singh Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-end training of neural retrievers for open-domain question answering. In *ACL/IJCNLP*.
- 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 *AAAI*.
- Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *EMNLP*, pages 9275–9293.
- Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *ICLR*.
- Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2022. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval. In *Findings of ACL*, pages 3557–3569.
- Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the use of lucene for information retrieval research. In *SIGIR*.
- Sohee Yang and Minjoon Seo. 2020. Is retriever merely an approximator of reader? *arXiv preprint arXiv:2010.10999*.
- Jingtao Zhan, Jiabin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing dense retrieval model training with hard negatives. In *SIGIR*.
- Jingtao Zhan, Jiabin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. 2020. Repbert: Contextualized text embeddings for first-stage retrieval. *arXiv preprint arXiv:2006.15498*.
- Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, and Weizhu Chen. 2021. Adversarial retriever-ranker for dense text retrieval. In *International Conference on Learning Representations*.
- Jiawei Zhou, Xiaoguang Li, Lifeng Shang, Lan Luo, Ke Zhan, Enrui Hu, Xinyu Zhang, Hao Jiang, Zhao Cao, Fan Yu, et al. 2022a. Hyperlink-induced pre-training for passage retrieval in open-domain question answering. In *ACL*, pages 7135–7146.
- Kun Zhou, Beichen Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2022b. Debaised contrastive learning of unsupervised sentence representations. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6120–6130.

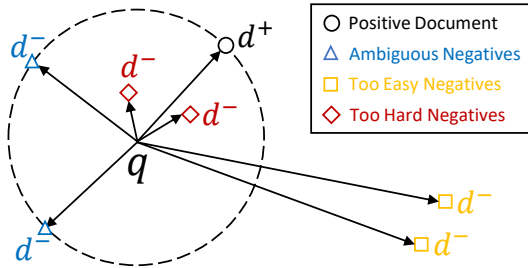


Figure 4: An example of the dense embedding distribution of a query with its positive document, too easy, too hard and ambiguous negatives.

A Illustration of Ambiguous Negatives

We illustrate the distribution of the dense embeddings of a query with its positive document, too easy, too hard and ambiguous negatives in Figure 4. Too hard negatives have a higher risk of being false negatives, and we can see that their dense embeddings locate closely to the ones of the query and the positive. If we learn to push them away, the distances between the embeddings of the query and the positive may also be enlarged, which is harmful to the goal of pulling the query and its positives together. Besides, too easy negatives locate rather far from the query, hence it is unnecessary to learn to push them even further. As a comparison, the ambiguous negatives have similar distances as the positive, which compose the *circular boundary* for the document pool consisting of hard negatives required to learn (*i.e.*, push away). In this way, our SimANS can be seen as always sampling the borderline hard negatives from the document pool. By learning to push them away, we can narrow the circular boundary of hard negatives, which helps gradually achieve the goal that pulls the query and positives together while pushing apart negatives.

B More Details on Datasets

We conduct experiments on five datasets, consisting of three passage retrieval datasets: Natural Question (NQ) (Kwiatkowski et al., 2019), Trivia QA (TQ) (Joshi et al., 2017) and MS-MARCO Passage Ranking (MS Pas) (Nguyen et al., 2016), a document retrieval dataset: MS-MARCO Document Ranking (MS Doc) (Nguyen et al., 2016) and a real-world industry dataset Bing. NQ and TQ are open domain question answering datasets collected from Google search logs and authored by trivia enthusiasts, respectively. In the two datasets, each question is paired with an answer span and sev-

eral golden passages from Wikipedia articles. Following existing works (Zhang et al., 2021; Sachan et al., 2021), we adopt Recall@k (R@k) as the evaluation metrics, which measures if the top- k ranked documents include the answer span. MS Pas and MS Doc consist of real questions collected from Bing search logs, where each question is paired with several web passages and documents, respectively. As their labels of test sets are not available, we follow existing works (Ren et al., 2021b; Zhan et al., 2021) that report results on their development sets and adopt MRR@10, R@50 and R@1k for MS Pas, MRR@10 and R@100 for MS Doc. Bing is collected from Bing search logs, where each example consists of a user historical query and several documents that the user has clicked. These documents are real-world webpages and may contain hyperlinks and different languages. We select Hit@5, Hit@20 and Hit@100 for evaluation.

C More Details on Baselines

We compare our approach with a variety of methods, including sparse and dense retrieval models.

- **BM25** (Yang et al., 2017) is a widely-used sparse retriever based on exact matching.
- **GAR** (Mao et al., 2021), **doc2query** (Nogueira et al., 2019a), **DeepCT** (Dai and Callan, 2019) and **docTTTTTquery** (Nogueira et al., 2019b) enhance BM25 by incorporating neural models.
- **DPR** (Karpukhin et al., 2020), **ANCE** (Xiong et al., 2021) and **STAR** (Zhan et al., 2021) are dense retrieval methods that adopt top- k hard negatives to improve training.
- **COIL** (Gao et al., 2021b) and **MEBERT** (Luan et al., 2021) combine sparse and dense representations for text retrieval.
- **Joint and Individual top- k** (Sachan et al., 2021) propose to train the dense retrieval model in an end-to-end manner.
- **RocketQA** (Qu et al., 2021), **RDR** (Yang and Seo, 2020), **RocketQAv2** (Ren et al., 2021b) and **ERNIE-search** (Lu et al., 2022) utilize knowledge distillation technique that leverages a teacher model to guide the training of the dense retrieval model.
- **PAIR** (Ren et al., 2021a), **DPR-PAQ** (Oğuz et al., 2022), **Condenser** (Gao and Callan, 2021) and **coCondenser** (Gao and Callan, 2022) design special pre-training tasks to improve the backbone model for the dense retrieval task.
- **AR2** (Zhang et al., 2021) incorporates an adversarial framework to jointly train the retriever

and the ranker. As it has achieved state-of-the-art performance on most datasets, we implement our approach on it to verify its effectiveness.

D Experimental Details

Implementation Details on Public Datasets. For three passage retrieval tasks, we follow the experimental settings in AR2 (Zhang et al., 2021) that selects ERNIE-2.0-base (Sun et al., 2020) as the backbone model. For MS Doc dataset, we leverage the model parameters of STAR (Zhan et al., 2021) to initialize AR2, and then train AR2 with the same hyper-parameters as STAR until convergence. Next, we continue to train the AR2 model parameters with our proposed SimANS, where we set a and b to $\{(0.5, 1.0), (0.5, 0), (0.5, 0), (0.5, 0)\}$ for NQ, TQ, MS Pas and MS Doc datasets, respectively. The learning rate is set to $1e-5$ for NQ and $5e-6$ for other datasets. The batch size is 256 for MS-Pas and MS-Doc, 64 for NQ and TQ, and the sampling ratio of positives and negatives is 1:15. All other hyper-parameter settings are the same as AR2. All the experiments in this work are conducted on 8 NVIDIA Tesla A100 GPUs.

Implementation Details on Bing Industry Dataset. For the industry dataset, Bing, we adopt mBERT-base (Devlin et al., 2019) as the backbone of the query and document encoders, to deal with multilingual queries and documents. The parameters of the baseline model are trained with randomly sampled negatives using the infoNCE loss (Karpukhin et al., 2020), namely **Baseline+Random Neg**, and the sampling ratio of positives and negatives is 1:5. The learning rate is $1e-5$, the batch size is 128 and the training step is 100,000. As a comparison, we implement the top- k negatives sampling strategy on the baseline model, namely **Baseline+top- k Neg**, where we utilize the baseline model to rank and select the top 5 documents that do not contain the query as hard negatives. In our approach, namely **Baseline+SimANS**, we continue to train the Baseline+top- k Neg model, but apply our SimANS to sample 5 negatives from the top 100 ranked documents. We set a to 1, b to 0, and reuse the other hyper-parameters of the Baseline+top- k Neg model.

E Case Study

In this part, we show four examples of the generated sampling probability distributions by our

SimANS. These four examples are randomly selected from the training set of MS Pas dataset. As shown in Figure 5, we can see that SimANS indeed assigns larger probabilities to the negatives that rank near the positive while punishing the higher-ranking and lower-ranking ones that may be false negatives and uninformative negatives. Furthermore, in Figure 5b where the positive is ranked at the first place, our approach is similar to the top- k negatives sampling method that assigns larger probabilities to the higher-ranking hard negatives.

F Related Work

Recent years have witnessed the remarkable performance of dense retrieval methods in text retrieval tasks (Zhan et al., 2020; Hong et al., 2022; Ram et al., 2022; Zhou et al., 2022b). Different from traditional sparse retrieval methods (e.g., TF-IDF and BM25), dense retrieval approaches typically map queries and documents into low-dimensional dense vectors, and then utilize vector distance metrics (e.g., cosine similarity) for retrieval.

To learn an effective dense retrieval model, it is key to sample high-quality negatives paired with the given query and positives for training. Early works (Karpukhin et al., 2020; Min et al., 2020) mostly rely on in-batch random negatives and hard negatives sampled by BM25. After that, a series of works (Qu et al., 2021; Xiong et al., 2021) find that sampling top- k ranked examples by the dense retriever as hard negatives is more helpful to improve the retriever itself. Among them, several methods (Xiong et al., 2021; Zhan et al., 2021) adopt a dynamic sampling strategy that actively samples top- k hard negatives once after an interval during training. However, these top- k negative sampling strategies are easy to select higher-ranking false negatives for training. To alleviate it, previous works have incorporated knowledge distillation (Qu et al., 2021; Ren et al., 2021b; Lu et al., 2022), pre-training (Zhou et al., 2022a; Xu et al., 2022) and other denoising techniques (Mao et al., 2022; Hofstätter et al., 2021). Despite the effectiveness, these methods mostly rely on complicated training strategies or complementary models.

In this work, we propose a simple but effective sampling method that weights the negative candidates with the consideration of their differences of relevance scores with positives. As a result, the ambiguous negatives with similar relevance scores to the positives will receive larger sampling probabili-

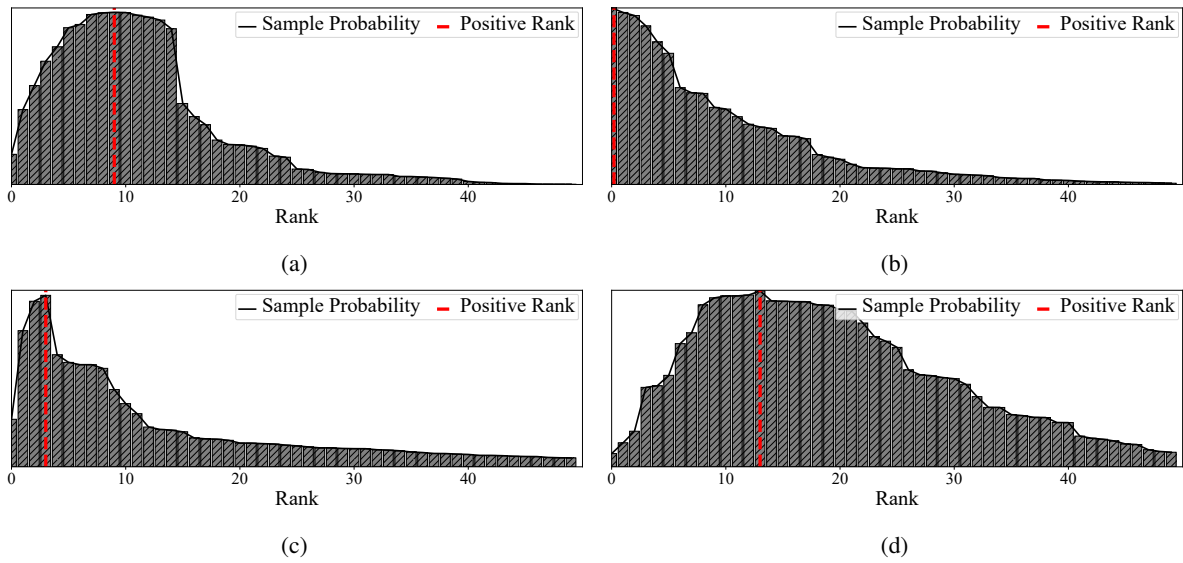


Figure 5: Illustration of four sampling probability distributions of the top 50 ranked negatives generated by our SimANS on the training set of MS Pas.

ties, while the too hard (potential false negatives) and too easy negatives (uninformative) will be punished with smaller probabilities.

Revisiting and Advancing Chinese Natural Language Understanding with Accelerated Heterogeneous Knowledge Pre-training

Taolin Zhang^{1,2}, Junwei Dong^{2,3}, Jianing Wang^{1,2}, Chengyu Wang^{2*}, Ang Wang², Yinghui Liu², Jun Huang², Yong Li², Xiaofeng He¹

¹ East China Normal University, Shanghai, China

² Alibaba Group, Hangzhou, China

³ Chongqing University, Chongqing, China

zhangtl0519@gmail.com, chengyu.wcy@alibaba-inc.com

Abstract

Recently, knowledge-enhanced pre-trained language models (KEPLMs) improve context-aware representations via learning from structured relations in knowledge graphs, and/or linguistic knowledge from syntactic or dependency analysis. Unlike English, there is a lack of high-performing open-source Chinese KEPLMs in the natural language processing (NLP) community to support various language understanding applications. In this paper, we revisit and advance the development of Chinese natural language understanding with a series of novel Chinese KEPLMs released in various parameter sizes, namely CKBERT (Chinese knowledge-enhanced BERT). Specifically, both relational and linguistic knowledge is effectively injected into CKBERT based on two novel pre-training tasks, i.e., linguistic-aware masked language modeling and contrastive multi-hop relation modeling. Based on the above two pre-training paradigms and our in-house implemented TorchAccelerator, we have pre-trained base (110M), large (345M) and huge (1.3B) versions of CKBERT efficiently on GPU clusters. Experiments demonstrate that CKBERT outperforms strong baselines for Chinese over various benchmark NLP tasks and in terms of different model sizes.¹

1 Introduction

Pre-trained Language Models (PLMs) such as BERT (Devlin et al., 2019) are pre-trained by self-supervised learning on large-scale text corpora to capture the rich semantic knowledge of words (Li et al., 2021; Gong et al., 2022), improving various downstream NLP tasks significantly (He et al., 2020; Xu et al., 2021; Chang et al., 2021). Although these PLMs have stored much internal knowledge (Petroni et al., 2019, 2020), they can

hardly understand external background knowledge from the world such as factual and linguistic knowledge (Colon-Hernandez et al., 2021; Cui et al., 2021; Lai et al., 2021).

In the literature, most approaches of knowledge injection can be divided into two categories, including relational knowledge and linguistic knowledge. (1) Relational knowledge-based approaches inject entity and relation representations in Knowledge Graphs (KGs) trained by knowledge embedding algorithms (Zhang et al., 2019; Peters et al., 2019) or convert triples into sentences for joint pre-training (Liu et al., 2020; Sun et al., 2020). (2) Linguistic knowledge-based approaches extract semantic units from pre-training sentences such as part-of-speech tags, constituent and dependency syntactic parsing, and feed all linguistic information into various transformer-based architectures (Zhou et al., 2020; Lai et al., 2021). We observe that there can be three potential drawbacks. (1) These approaches generally utilize a single source of knowledge (i.e., inherent linguistic knowledge), which ignore important knowledge from other sources (Su et al., 2021) (i.e., relational knowledge from KGs). (2) Training large-scale KEPLMs from scratch requires high-memory computing devices and is time-consuming, which brings significant computational burdens for users (Zhang et al., 2021, 2022). (3) Most of these models are pre-trained in English only. There is a lack of powerful KEPLMs for understanding other languages (Lee et al., 2020; Pérez et al., 2021).

To overcome the above problems, we release a series of Chinese KEPLMs named CKBERT (Chinese knowledge-enhanced BERT), with heterogeneous knowledge sources injected. We particularly focus on Chinese as it is one of the most widely spoken languages other than English. The CKBERT models are pre-trained by two well-designed pre-training tasks as follows:

- **Linguistic-aware Masked Language Mod-**

* Corresponding author.

¹All the codes and model checkpoints have been released to public in the EasyNLP framework (Wang et al., 2022). URL: <https://github.com/alibaba/EasyNLP>.

eling (LMLM): LMLM is substantially extended from Masked Language Modeling (MLM) (Devlin et al., 2019) by introducing two key linguistics tokens derived from dependency syntactic parsing and semantic role labeling. We also insert unique markers for each linguistic component among contiguous tokens. The goal of LMLM is to predict both randomly selected tokens and linguistic tokens masked in the pre-training sentences.

- **Contrastive Multi-hop Relation Modeling (CMRM):** We sample fine-grained subgraphs from a large-scale Chinese KG by multi-hop relations to compensate for understanding the background knowledge of target entities. Specifically, we construct positive triples for matched target entities via retrieving one-hop entities in the corresponding subgraphs. Negative triples are sampled from unrelated multi-hop entities through the relation paths in the KG. The CMRM task is proposed to pull the semantics of similar entities close and push away those with irrelevant semantics.

Based on the above heterogeneous knowledge pre-training tasks, we produce various sizes of CKBERT models to meet the inference time and accuracy requirements of different real-world scenarios (Brown et al., 2020; Chowdhery et al., 2022), including base (110M), large (345M) and huge (1.3B). The models are pre-trained using our in-house implemented TorchAccelerator that effectively transforms PyTorch eager execution to graph execution on distributed GPU clusters, boosting the training speed by 40% per sample with our advanced compiler technique based on Accelerated Linear Algebra (XLA). In the experiments, we compare CKBERT against strong baseline PLMs and KEPLMs on various Chinese general and knowledge-related NLP tasks. The results demonstrate the improvement of CKBERT compared to SoTA models.

2 Related Work

We briefly summarize the related work on the following two aspects: PLMs and KEPLMs.

2.1 PLMs

Following BERT (Devlin et al., 2019), many PLMs have been proposed to improve performance in various NLP tasks. Several approaches extend BERT

by employing novel token-level and sentence-level pre-training tasks. Notable PLMs include ERNIE-Baidu (Sun et al., 2019), MacBERT (Cui et al., 2020) and PERT (Cui et al., 2022) for Chinese NLU downstream tasks. Other models boost the performance by changing the internal encoder architectures. For example, XLNet (Yang et al., 2019) utilizes Transformer-XL (Dai et al., 2019) to encode long sequences by the permutation in language tokens. Sparse self-attention (Cui et al., 2019) replaces the self-attention mechanism with more interpretable attention units. Yet, other PLMs such as MT-DNN (Liu et al., 2019) combine self-supervised pre-training with the multi-task supervised learning to improve the performance of various GLUE tasks (Wang et al., 2019).

2.2 KEPLMs

These models use structured knowledge or linguistic semantics to enhance the language understanding abilities of PLMs. We summarize recent KEPLMs grouped into the following four types: (1) Knowledge-enhancement by linguistic semantics. These works use the linguistic information already available in the pre-training sentences to enhance the understanding ability of PLMs. Lattice-BERT (Lai et al., 2021) pre-trains a Chinese PLM over a word lattice (Buckman and Neubig, 2018) structure to exploit multi-granularity inputs. (2) Knowledge-enhancement by entity embeddings. For example, ERNIE-THU (Zhang et al., 2019) injects entity embeddings into contextual representations via knowledge-encoders stacked by the information fusion module. (3) Knowledge-enhancement by entity descriptions. These approaches learn entity embeddings by knowledge descriptions. For example, pre-training corpora and entity descriptions in KEPLER (Wang et al., 2021) are encoded into a unified semantic space within the same PLM. (4) Knowledge-enhancement by converted triplet’s texts. K-BERT (Liu et al., 2020) and CoLAKE (Sun et al., 2020) convert relation triplets into texts and insert them into training samples without using pre-trained embeddings. In this paper, we argue that aggregating heterogeneous knowledge information can further benefit the context-aware representations of PLMs.

3 Model

In this section, we elaborate the techniques of the proposed CKBERT model. The main architecture

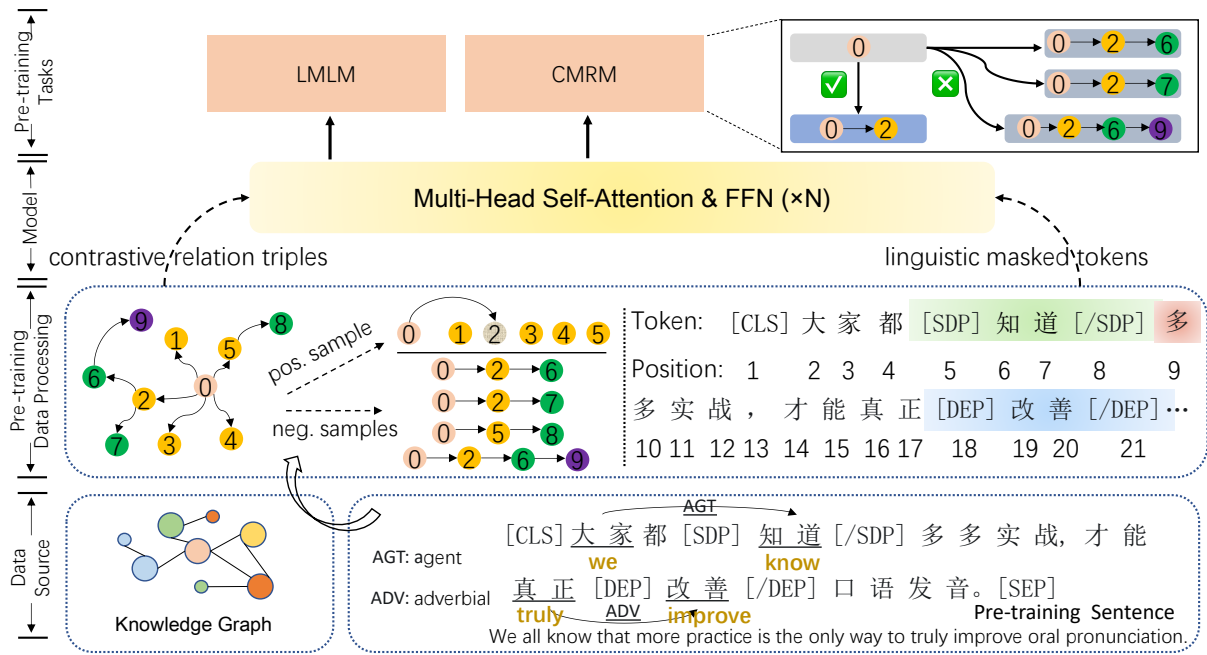


Figure 1: Model overview. The LMLM task is not only able to perform random masked token prediction (similar to BERT) but also to predict masked linguistic-aware tokens. The CMRM task injects external relation triples into PLMs through neighboring multi-hop relations. (Best viewed in color.)

of CKBERT is firstly presented in Figure 1.

3.1 Model Architecture

It accepts a sequence of M WordPiece tokens (Wu et al., 2016), (x_1, x_2, \dots, x_M) as input, and computes the D -dimensional contextual representations $H_i \in \mathbb{R}^{M \times D}$ by successively stacking N transformer encoder layers. We do not modify the architecture here to guarantee that CKBERT can be seamlessly integrated into any industrial applications that BERT supports with better performance.²

3.2 Linguistic-aware Masked Language Modeling (LMLM)

In BERT pre-training, 15% of all token positions are randomly masked for prediction. However, random masked tokens may be unimportant units such as conjunctions and prepositions (Clark et al., 2019; Hao et al., 2021). We reconstruct the input sentences and mask more tokens based on linguistic knowledge so that CKBERT can better understand the semantics of important tokens in pre-training sentences. Specifically, we use the following three steps to mask the linguistic input units:

- **Recognizing Linguistic Tokens:** We first use

²Without loss of generality, we focus on the transformer encoder architecture only; yet our work can also be extended model architectures with slight modification.

the off-the-shelf tool³ to recognize important units in pre-training sentences, including dependence grammar and semantic dependency parsing. The extracted relations here serve as important sources of linguistic knowledge, including “subject-verb”, “verb-object” and “adverbial” for dependence grammar and “non-agent” for semantic dependency parsing.

- **Reconstructing Input Sentences:** In addition to the original input form, based on the subjects and objects of the extracted linguistic relations, we insert special identifiers for each lexicon unit between words spans to give explicit boundary information for model pre-training. For example, we add [DEP] and [/DEP] for dependence grammar and [SDP] and [/SDP] for dependency parsing tokens.
- **Choosing Masked Tokens:** We choose 15% of token positions from the reconstructed input sentence for masking, using the special token [MASK]. Among these tokens, we assign 40% of the positions to randomly selected tokens and the rest to linguistic tokens. Note that these special identifiers ([DEP], [/DEP], [SDP] and [/SDP]) are also treated as normal tokens for masking, thus the model needs to

³<http://ltp.ai/>

be aware of predicting word boundaries rather than simply filling in masks based on contexts.

After input sentences are processed, for LMLM, let $\Omega = (m_1, m_2, m_3, \dots, \gamma_{K-1}, \gamma_K)$ denote the indexes of the masked tokens in the sentence X , where m_i is an index of a randomly masked token, γ_i is an index of a selected linguistic-aware masked token and K is the total number of masked tokens. Let X_Ω denote the set of masked tokens in X , and $X_{-\Omega}$ denote the set of observed (unmasked) tokens. The objective of LMLM is as follows:

$$\mathcal{L}_{mlm}(X_\Omega|X_{-\Omega}) = \frac{1}{K} \sum_{k=1}^K \log p(x_{m_k|\gamma_k}|X_{-\Omega}; \theta) \quad (1)$$

where $x_{m_k|\gamma_k}$ denotes the randomly selected tokens or the linguistic tokens. θ represents the parameter collection of our model.

3.2.1 Contrastive Multi-hop Relation Modeling (CMRM)

In addition to LMLM, we further inject relation triples into CKBERT to make it understand the background factual knowledge of entities. For an entity in the pre-training sentence, we construct positive and negative relation triples as follows:

- **Positive Triples:** We employ entity linking to link an entity in the pre-training sentence to the target entity e_t in the KG. The relation triples w.r.t. the one-hop entities are viewed as candidate positive triples. Next, we choose a relation triple randomly from the candidates as a positive sample, denoted as t_p .
- **Negative Triples:** Because the semantic similarity between the positive triple t_p and the relation triples along the KG paths decreases, we construct L candidate negative triples $(t_n^1, t_n^2, \dots, t_n^L)$ by making multiple hops starting from the target entity e_t . For example, in Fig. 1, we take the target entity e_0 as the starting node and retrieve the nodes along the edges. We obtain the ending node e_{end} with multi-hop relations $Hop(\mathcal{G}, e_0, e_{end}, r)$, where $Hop(\cdot)$ means the shortest distance between e_0 and e_{end} in KG \mathcal{G} . Here, we regard a triple to be negative t_n if $Hop(\cdot) > 1$ and is no larger than a small threshold δ^4 . In this

⁴If the threshold for the number of hops δ is too large, the model can easily distinguish the positive and negative triples due to the large semantic gaps. For effective contrastive learning, good negative triples should be ‘‘hard negatives’’.

paper, we set $\delta = 3$. Hence, there are four negative triples for e_0 in Figure 1. A sample three-hop path is $e_0 \rightarrow e_2 \rightarrow e_6 \rightarrow e_9$.

The CMRM task is designed for pulling similar relational triples of the target entity closely and pushing unrelated multi-hop relational triples away, in order to enhance the external background knowledge of the target entity from the KG. Concretely, after the positive sample t_p and negative samples $(t_n^1, t_n^2, \dots, t_n^L)$ of the target entity e_t are retrieved, the context-aware representations of the target entity e_t can be obtained as follows:

$$h_{e_t} = \mathcal{LN} \left(\sigma \left(f_{sp} \left(h_{e_t^i}, \dots, h_{e_t^j} \right) W_1 \right) \right) \quad (2)$$

where h_{e_t} is the hidden representation of the target entity e_t constructed by the entity’s token representations $(h_{e_t^i}, \dots, h_{e_t^j})$, as an entity can have multiple tokens in the pre-training sentence. f_{sp} is the self-attentive pooling operator (Lin et al., 2017), $\sigma(\cdot)$ non-linear activation function GELU (Hendrycks and Gimpel, 2016) and $\mathcal{LN}(\cdot)$ is the LayerNorm function (Ba et al., 2016). W_1 is the learnable weight matrix.

Meanwhile, as relation triples can be viewed as natural sentences via concatenating the triple’s tokens together, following Liu et al. (2020); Sun et al. (2020), we convert the triples into sentences to generate the representations obtained by the shared encoder θ (which is the transformer encoder of our CKBERT model). Hence, the representations of the positive triple h_{t_p} and the negative triples $(h_{t_n^1}, h_{t_n^2}, \dots, h_{t_n^L})$ can also be derived. For the CMRM task, we employ InfoNCE (van den Oord et al., 2018) as the loss function to calculate the similarity as follows:

$$\mathcal{L}_{cl} = -\log \frac{\exp(\cos(h_{e_t}, h_{t_p})/\tau)}{\sum_{l=1}^L \exp(\cos(h_{e_t}, h_{t_n^l})/\tau)} \quad (3)$$

where $\cos(\cdot, \cdot)$ denotes the cosine function to calculate the similarity between entity and relation representations, and τ is a pre-defined hyper-parameter.

3.3 Optimization of Model Pre-training

For model training optimization, we first give the total loss function for pre-training CKBERT based on our two novel pre-training tasks as follows:

$$\mathcal{L}_{total} = \mathcal{L}_{mlm} + \mathcal{L}_{cl} \quad (4)$$

Model	Text Classification						Question Answering			Total
	AFQMC	TNEWS	IFLYTEK	OCNLI	WSC	CSL	CMRC	CHID	C3	Score
BERT	72.73	55.22	59.54	66.53	72.49	81.77	73.40	79.19	57.91	69.72
MacBERT	69.90	57.93	60.35	67.43	74.71	82.13	73.55	79.51	58.89	70.28
PERT	73.61	54.50	57.42	66.70	76.07	82.77	73.80	80.19	58.03	70.18
ERNIE-Baidu	73.08	56.22	60.11	67.48	75.79	82.14	72.86	80.03	57.63	69.83
Lattice-BERT	72.96	56.14	58.97	67.54	76.10	81.99	73.47	80.24	57.80	70.29
K-BERT	73.15	55.91	60.19	67.83	76.21	82.24	72.74	80.29	57.48	70.35
ERNIE-THU	72.88	56.59	59.33	67.95	75.82	82.35	72.96	80.22	56.30	69.98
CKBERT-base	73.17	56.44	60.65	68.53	76.38	82.63	73.55	81.69	57.91	71.36
CKBERT-large	74.75	55.86	60.62	70.57	78.90	82.30	73.45	82.34	58.12	72.23
CKBERT-huge	75.03	59.72	60.96	78.26	85.16	89.47	77.25	97.73	86.59	78.91
CKBERT-huge (Ensemble)	77.05	61.16	61.19	82.80	87.14	94.23	80.40	97.91	87.26	81.02

Table 1: Performance of tasks on the CLUE 1.1 testing sets (%). The ‘‘Total Score’’ is the weighted averaged score of the nine tasks generated by the official website automatically. All the baseline models are base models (with the same or similar parameter size as that of BERT-base).

Here, we pre-train a series of CKBERT models on distributed GPU clusters, with codes in PyTorch. As PyTorch employs eager execution for tensor computation, it lacks graph-based intermediate representations of models, hindering deeper optimization (Paszke et al., 2019).

Inspired by LazyTensor (Suhan et al., 2021) and Pytorch/XLA on cloud TPUs⁵, we develop the TorchAccelerator toolkit for Pytorch training acceleration on GPU clusters. Through XLA custom function and code parsing with an abstract syntax tree (AST), we improve the completeness and performance of the transformation from eager execution to graph execution. A computational graph is generated by TorchAccelerator. The operators on the graph will be fused. By fusing operators, the kernel launch overhead can be reduced. Moreover, fewer intermediate results are written to memory thus reducing the memory bandwidth usage. The effectiveness of computation is also improved by multi-stream optimization and asynchronous transmission of tensors. Since the implementation of TorchAccelerator is not our major focus, more details will be presented in our future work.

4 Experiments

We present comprehensive evaluation results of CKBERT. Due to space limitation, the details of data sources, baselines and hyper-parameter settings are shown in Appendices A, B, C.

⁵<https://github.com/pytorch/xla>

4.1 General Experimental Results

We evaluate CKBERT over a widely-used Chinese benchmark CLUE (Xu et al., 2020) and knowledge-intensive tasks to evaluate the influence of knowledge injection in CKBERT.

4.1.1 Results of CLUE Benchmark

The CLUE benchmark contains nine text classification and question answering tasks. Specifically, the text classification tasks contain various text task types, including the classification of short sentences and long sentence pairs. The results of all tasks are shown in Table 1.

From the results, we have the following observations. (1) The performance of KEPLMs has a large gap over BERT. It indicates that the injection of different knowledge sources enables the models to perform better semantic reasoning compared to pre-training on texts only. (2) The performance of CKBERT is further improved compared to previous strong baseline KEPLMs under the same parameter size in most cases. From this phenomenon, we believe that the heterogeneous knowledge sources injected into the PLMs benefit the model’s results. (3) The larger the number of parameters in the model, the more effective the heterogeneous knowledge fusion is for downstream tasks. The huge model of CKBERT (1.3B parameters) outperforms base (110M) by a large margin, which is suitable for applications that require high prediction accuracy. We also build an ensemble of the huge models (denoted as CKBERT-huge (Ensemble)) from different checkpoints. The performance can be further improved by more than 2.0%.

Model	MSRA	Weibo	Onto.	Resu.
BERT	95.20	54.65	81.61	94.86
MacBERT	95.07	54.93	81.96	95.22
PERT	94.99	53.74	81.44	95.10
ERNIE-BD	95.39	55.14	81.17	95.13
Lat.-BERT	95.28	54.99	82.01	95.31
K-BERT	94.97	55.21	81.98	94.92
ERNIE-THU	95.25	53.85	82.03	94.89
CKBERT-base	95.35	55.97	82.19	95.68
CKBERT-large	95.98	57.09	82.43	96.08
CKBERT-huge	96.79	58.66	83.87	97.19

Table 2: Performance of CKBERT and baselines over four public Chinese NER datasets in term of F1 (%).

4.1.2 Results of NER

We further evaluate CKBERT over the four public NER datasets, including MSRA (Levow, 2006), Weibo (Peng and Dredze, 2015), Ontonotes 4.0⁶, and Resume (Yang et al., 2017). The detailed statistics including the split sizes of training, development, and testing sets are described in Appendix A. The models are stacked by the CKBERT encoder and a softmax linear layer, whose parameters are initialized randomly. The entities recognized in the samples are labeled by the B/I/O/S tags. This transforms the NER task into a 4-class classification task for each token.

Table 2 shows the performance of various models on four NER datasets. It can be seen that KEPLMs outperform vanilla PLMs. In addition, our CKBERT model (base) with linguistic and external knowledge achieves a large gap performance compared to baselines. We believe that heterogeneous knowledge sources play an important role as described in the ablation study (See Section 4.2).

4.2 Ablation Study

In this part, we evaluate the effectiveness of two important model components of CKBERT on representative tasks. Specifically, We introduce several variants of CKBERT removing certain components. CKBERT-LMLM means that we remove the LMLM task and only learns the CMRM task during pre-training. CKBERT-CMRM remove the CRMR task and only perform the LMLM task. We also provide the results of continual pre-training of BERT-base to remove the influence of additional data sources of plain texts. The performance of those variants and CKBERT on the testing sets of these datasets are shown in Table 3.

⁶<https://catalog.ldc.upenn.edu/LDC2013T19>

Model	AFQ.	IFLY.	CMRC	Weibo
BERT-large-con.	73.96	60.35	73.42	56.12
CKBERT-large	74.75	60.62	73.45	57.09
w/o. LMLM	73.56	60.38	73.2	56.48
w/o. CMRM	72.96	59.38	74.8	56.72

Table 3: The performance of models for ablation study. “AFQ.” and “IFLY.” refer to AFQMC and IFLYTEX, respectively (%).

From the results, we can see that (1) Comparing CKBERT-large to BERT-large (continual pre-trained with the same pre-training data), the explicit heterogeneous knowledge is more useful than the implicit text corpus for various downstream tasks. (2) We also find that the LMLM pre-training task benefits the QA and NER tasks more, whereas the CMRM task improves the performance of plain NLU task (i.e., text classification) significantly. We conjecture that the main reason behind this phenomenon is that the external background knowledge can easily boost the performance due to the shallow semantics of these simple tasks (Yang et al., 2021).

4.3 Results of TorchAccelerator

We investigate to what extent the pre-training speed is improved when our framework is integrated with TorchAccelerator. Figure 2 shows the comparison results between TorchAccelerator and Torch Native with AMP (Automatic Mixed Precision) ⁷. The metric “samples/s” means how many samples are computed by the model in each second. Note that we increase the batch size as large as possible to increase the GPUs’ memory utilization and occupancy to 100%, and thus the experiments w/ and w/o. TorchAccelerator consume the same amount of computational resources. The underlying GPU is Tesla V100 32GB.

From the results, our observations are as follows. (1) When we only use TorchAccelerator without AMP, the training speed increases slightly. (2) The training speed can have a large improvement with the interaction between TorchAccelerator and AMP (+40%). This is because the kernel fusion of XLA in TorchAccelerator largely reduces the amount of memory access operations, which are the performance bottleneck when AMP is applied. Hence, our TorchAccelerator effectively reduces the consumption of resources and time during pre-training.

⁷<https://github.com/NVIDIA/apex>

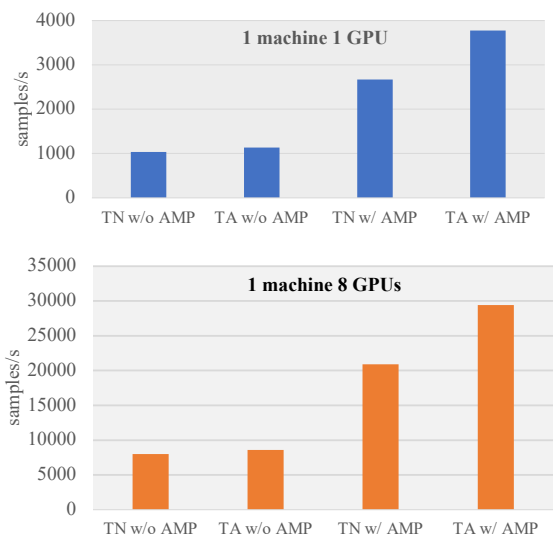


Figure 2: Training speed comparison between TorchAccelerator (TA) and Torch Native (TN).

5 Conclusion and Future Work

In this paper, we propose a novel series of Chinese KEPLMs named CKBERT to inject the heterogeneous sources including linguistic and external knowledge into the PLMs. Specifically, we design two novel pre-training tasks including linguistic-aware MLM and contrastive multi-hop relation modeling, and accelerate model pre-training by TorchAccelerator. The experiments show that our CKBERT outperforms various strong baselines including general PLMs and KEPLMs significantly over knowledge-intensive and natural language understanding tasks. Future work includes (1) integrating more knowledge sources into PLMs to further improve the performance of downstream tasks; (2) exploring heterogeneous knowledge injection to generative KEPLMs and other languages; and (3) enriching the functionalities of TorchAccelerator and releasing it to public.

Ethical Considerations

Our contribution in this work is fully methodological, namely a novel series of KEPLMs, achieving the performance improvement of downstream tasks with different parameter sizes. Hence, there is no explicit negative social influences in this work. However, transformer-based models may have some negative impacts, such as gender and social bias. Our work would unavoidably suffer from these issues. We suggest that users should carefully address potential risks when the CKBERT models are deployed online.

Acknowledgments

This work has been supported by Alibaba Group through Alibaba Innovative Research Program and Alibaba Research Intern Program.

References

- Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. [Layer normalization](#). *CoRR*, abs/1607.06450.
- 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](#). In *NeurIPS*.
- Jacob Buckman and Graham Neubig. 2018. [Neural lattice language models](#). *Trans. Assoc. Comput. Linguistics*, 6:529–541.
- Tyler A. Chang, Yifan Xu, Weijian Xu, and Zhuowen Tu. 2021. [Convolutions and self-attention: Re-interpreting relative positions in pre-trained language models](#). In *ACL*, pages 4322–4333.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pilla, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. [Palm: Scaling language modeling with pathways](#). *CoRR*, abs/2204.02311.
- Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. [What does BERT look at? an analysis of bert’s attention](#). In *ACL*, pages 276–286.

- Pedro Colon-Hernandez, Catherine Havasi, Jason B. Alonso, Matthew Huggins, and Cynthia Breazeal. 2021. [Combining pre-trained language models and structured knowledge](#). *CoRR*, abs/2101.12294.
- Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. [Fine-tune BERT with sparse self-attention mechanism](#). In *EMNLP*, pages 3546–3551.
- Leyang Cui, Sijie Cheng, Yu Wu, and Yue Zhang. 2021. [On commonsense cues in BERT for solving commonsense tasks](#). In *ACL*, pages 683–693.
- Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. [Revisiting pre-trained models for chinese natural language processing](#). In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 657–668. Association for Computational Linguistics.
- Yiming Cui, Ziqing Yang, and Ting Liu. 2022. [PERT: pre-training BERT with permuted language model](#). *CoRR*, abs/2203.06906.
- 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 *ACL*, 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 *NAACL-HLT*, pages 4171–4186.
- Zheng Gong, Kun Zhou, Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2022. [Continual pre-training of language models for math problem understanding with syntax-aware memory network](#). In *ACL*, pages 5923–5933.
- Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2021. [Self-attention attribution: Interpreting information interactions inside transformer](#). In *AAAI*, pages 12963–12971.
- Yun He, Ziwei Zhu, Yin Zhang, Qin Chen, and James Caverlee. 2020. [Infusing disease knowledge into BERT for health question answering, medical inference and disease name recognition](#). In *EMNLP*, pages 4604–4614.
- Dan Hendrycks and Kevin Gimpel. 2016. [Gaussian error linear units \(gelus\)](#). *arXiv:1606.08415*.
- Yuxuan Lai, Yijia Liu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2021. [Lattice-bert: Leveraging multi-granularity representations in chinese pre-trained language models](#). In *NAACL*, pages 1716–1731.
- Hyunjae Lee, Jaewoong Yoon, Bonggyu Hwang, Seongho Joe, Seungjai Min, and Youngjune Gwon. 2020. [Korealbert: Pretraining a lite BERT model for korean language understanding](#). In *ICPR*, pages 5551–5557.
- Gina-Anne Levow. 2006. [The third international chinese language processing bakeoff: Word segmentation and named entity recognition](#). In *SIGHAN@COLING/ACL*, pages 108–117.
- Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. 2021. [How is BERT surprised? layer-wise detection of linguistic anomalies](#). In *ACL*, pages 4215–4228.
- Zhouhan Lin, Minwei Feng, Cícero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. [A structured self-attentive sentence embedding](#). In *ICLR*.
- Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. [K-BERT: enabling language representation with knowledge graph](#). In *AAAI*, pages 2901–2908.
- Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. [Multi-task deep neural networks for natural language understanding](#). In *ACL*, pages 4487–4496.
- 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 Z. 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 *NeurIPS*, pages 8024–8035.
- Nanyun Peng and Mark Dredze. 2015. [Named entity recognition for chinese social media with jointly trained embeddings](#). In *EMNLP*, pages 548–554.
- Juan Manuel Pérez, Damián Ariel Furman, Laura Alonso Alemany, and Franco Luque. 2021. [Robertuito: a pre-trained language model for social media text in spanish](#). *CoRR*, abs/2111.09453.
- Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. [Knowledge enhanced contextual word representations](#). In *EMNLP*, pages 43–54.
- Fabio Petroni, Patrick S. H. Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. [How context affects language models’ factual predictions](#). In *AKBC*.
- Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. [Language models as knowledge bases?](#) In *EMNLP*, pages 2463–2473.
- Yusheng Su, Xu Han, Zhengyan Zhang, Yankai Lin, Peng Li, Zhiyuan Liu, Jie Zhou, and Maosong Sun. 2021. [Cokebert: Contextual knowledge selection and embedding towards enhanced pre-trained language models](#). *AI Open*, 2:127–134.

- Alex Suhan, Davide Libenzi, Ailing Zhang, Parker Schuh, Brennan Saeta, Jie Young Sohn, and Denys Shabalin. 2021. Lazytensor: combining eager execution with domain-specific compilers. *CoRR*, abs/2102.13267.
- Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. 2020. Colake: Contextualized language and knowledge embedding. In *COLING*, pages 3660–3670.
- Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. ERNIE: enhanced representation through knowledge integration. *CoRR*, abs/1904.09223.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.
- 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 *ICLR*.
- Chengyu Wang, Minghui Qiu, Taolin Zhang, Tingting Liu, Lei Li, Jianing Wang, Ming Wang, Jun Huang, and Wei Lin. 2022. EasyNLP: A comprehensive and easy-to-use toolkit for natural language processing. *CoRR*, abs/2205.00258.
- Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Trans. Assoc. Comput. Linguistics*, 9:176–194.
- 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.
- Liang Xu, Hai Hu, Xuanwei Zhang, Lu Li, Chenjie Cao, Yudong Li, Yechen Xu, Kai Sun, Dian Yu, Cong Yu, Yin Tian, Qianqian Dong, Weitang Liu, Bo Shi, Yiming Cui, Junyi Li, Jun Zeng, Rongzhao Wang, Weijian Xie, Yanting Li, Yina Patterson, Zuoyu Tian, Yiwen Zhang, He Zhou, Shaowei Hua, Zhe Zhao, Qipeng Zhao, Cong Yue, Xinrui Zhang, Zhengliang Yang, Kyle Richardson, and Zhenzhong Lan. 2020. CLUE: A chinese language understanding evaluation benchmark. In *COLING*, pages 4762–4772.
- Zenan Xu, Daya Guo, Duyu Tang, Qinliang Su, Linjun Shou, Ming Gong, Wanjun Zhong, Xiaojun Quan, Daxin Jiang, and Nan Duan. 2021. Syntax-enhanced pre-trained model. In *ACL*, pages 5412–5422.
- Jian Yang, Gang Xiao, Yulong Shen, Wei Jiang, Xinyu Hu, Ying Zhang, and Jinghui Peng. 2021. A survey of knowledge enhanced pre-trained models. *CoRR*, abs/2110.00269.
- Jie Yang, Yue Zhang, and Fei Dong. 2017. Neural reranking for named entity recognition. In *RANLP*, pages 784–792.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NIPS*, pages 5754–5764.
- Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. 2022. DKPLM: decomposable knowledge-enhanced pre-trained language model for natural language understanding. In *AAAI*, pages 11703–11711.
- Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. In *ACL*, pages 1441–1451.
- Zhengyan Zhang, Xu Han, Hao Zhou, Pei Ke, Yuxian Gu, Deming Ye, Yujia Qin, Yusheng Su, Haozhe Ji, Jian Guan, Fanchao Qi, Xiaozhi Wang, Yanan Zheng, Guoyang Zeng, Huanqi Cao, Shengqi Chen, Daixuan Li, Zhenbo Sun, Zhiyuan Liu, Minlie Huang, Wentao Han, Jie Tang, Juanzi Li, Xiaoyan Zhu, and Maosong Sun. 2021. CPM: A large-scale generative chinese pre-trained language model. *AI Open*, 2:93–99.
- Junru Zhou, Zhuosheng Zhang, Hai Zhao, and Shuailiang Zhang. 2020. LIMIT-BERT: Linguistics informed multi-task BERT. In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 4450–4461.

A Data Statistics

A.1 Data Sources for Pre-training

The pre-training corpora after pre-processing contain 5 million text segments with 623,366,851 tokens (6.2 GB). We also perform simple data pre-processing on the these corpora to improve the quality of the data, including removing incorrect characters and non-Chinese characters, etc. Our KG data is downloaded from the largest authoritative Chinese KG website OpenKG⁸. The number of entities and triples of OpenKG are 16,474,936 and 140,883,574, respectively. The total number of relation types is 480,882.

A.2 Statistics of Downstream Tasks

In this paper, we choose three types downstream tasks for evaluation, including Text Classification (TC), Question Answering (QA) and Named Entity

⁸<http://openkg.cn/>

Dataset	# Train	# Dev	# Test	Task	Metric
AFQMC	34,334	4,316	3,861	TC	Acc@1
TNEWS	53,360	10,000	10,000	TC	Acc@1
IFLYTEK	12,133	2,599	2,600	TC	Acc@1
OCNLI	50,000	3,000	3,000	TC	Acc@1
WSC	1,244	304	2573	TC	Acc@1
CSL	20,000	3,000	3,000	TC	Acc@1
CMRC	10,142	1,002	3,219	QA	F1
CHID	84,709	3,218	3,231	QA	Acc@1
C3	11,869	3,816	3,892	QA	Acc@1
MSRA	40,000	6,675	4364	NER	F1
Resume	3,821	462	476	NER	F1
Weibo	1,350	264	262	NER	F1
Ontonotes	15,740	4300	4345	NER	F1

Table 4: The data statistics and evaluation metrics used in the experiments.

Model	$n_{param.}$	N_{layer}	N_{head}	D_{head}	D_{ff}	D_{model}
CKBERT-base	110M	12	12	64	3072	768
CKBERT-large	345M	24	16	64	4096	1024
CKBERT-huge	1.3B	24	8	256	8192	2048

Table 5: The overview of hyper-parameters settings of our model architectures. $n_{param.}$ means the total parameters of our model. N_{layer} is the number of model layers. N_{head} is the number of attention heads in each layer. D_{head} is the hidden dimension of attention heads. D_{ff} is the intermediate dimension of FFN layers. D_{model} is the output dimension of the model.

Recognition (NER). The statistics of dataset sizes are shown in Table 4. The result metrics used in our models are different among tasks. We use the Acc@1 for text classification, F1 for NER. For QA tasks, since CHID and C3 tasks are multiple choices, we use Acc@1 as the metric for the two tasks and F1 for CMRC.

B Baselines

In this work, we compare CKBERT with general PLMs and KEPLMs with knowledge triples injected, pre-trained on our text corpora:

B.1 General PLMs

We use three strong Chinese BERT-style models as baselines, namely BERT-base (Devlin et al., 2019), MacBERT (Cui et al., 2020) and PERT (Cui et al., 2022). All the model weights are initialized from

Cui et al. (2020).

B.2 KEPLMs

We employ three SoTA KEPLMs continually pre-trained on our pre-training corpora as our baseline models, including ERNIE-Baidu (Sun et al., 2019), ERNIE-THU (Zhang et al., 2019) and K-BERT (Liu et al., 2020). For a fair comparison, KEPLMs using other resources rather than the KG triples are excluded in this work. All the baseline KEPLMs are injected by the same KG triples during pre-training.

C Hyper-parameters Settings

C.1 Hyper-parameters of Pre-training

For optimization, we set the learning rate as $5e-5$, the max sequence length as 128, and the batch size as 20. The hidden dimension of the text encoder is

Dataset	AFQMC	TNEWS	IFLY.	OCNLI	WSC	CSL	CMRC
BS	4	12	4	32	32	16	16
Epoch	10	10	10	50	50	10	10
LR	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5
MSL	256	128	256	128	128	256	512
Dataset	CHID	C3	MSRA	Resume	Weibo	Ontonotes	
BS	4	4	32	32	32	32	
Epoch	15	15	10	10	10	10	
LR	5e-5	5e-5	5e-5	5e-5	5e-5	5e-5	
MSL	192	512	128	128	128	128	

Table 6: The important fine-tuning hyper-parameters used in our CKBERT models. “BS”, “LR”, and “MSL” indicate the batch size, the learning rate and the max sequence length, respectively.

768. The temperature hyper-parameter τ is set to 0.5. The number of negative samples L is 3. During pre-training, all the experiments are conducted on 15 servers, each with 8 Tesla V100 GPUs (32GB).

C.2 Hyper-parameters of Model Architectures

Table 5 shows the hyper-parameters settings of our CKBERT models w.r.t. the model architectures, including base (110M), large (345M) and huge (1.3B).

C.3 Hyper-parameters of Fine-tuning

Table 6 shows the hyper-parameters settings for fine-tuning. For fair comparison, we set a unified set of important hyper-parameters for each task, including the batch size, the learning epoch, the learning rate and the max sequence length.

A Stacking-based Efficient Method for Toxic Language Detection on Live Streaming Chat

Yuto Oikawa^{1,2}

Yuki Nakayama²

Koji Murakami²

¹Graduate School of Engineering, Kitami Institute of Technology

²Rakuten Institute of Technology, Rakuten Group Inc.

Abstract

In a live streaming chat on a video streaming service, it is crucial to filter out toxic comments with online processing to prevent users from reading comments in real-time. However, recent toxic language detection methods rely on deep learning methods, which can not be scalable considering inference speed. Also, these methods do not consider constraints of computational resources expected depending on a deployed system (e.g., no GPU resource). This paper presents an efficient method for toxic language detection that is aware of real-world scenarios. Our proposed architecture is based on partial stacking that feeds initial results with low confidence to meta-classifier. Experimental results show that our method achieves a much faster inference speed than BERT-based models with comparable performance.

1 Introduction

With the rapid growth of online social platforms, posting text comments to a content have become a familiar part of people’s lives for growing human connection and advertising purposes. However, these comments can include toxic language, which can be harmful or offensive to others. Toxic comments lead to damages of the user experience, human well-being, and even product promotion. Particularly, toxic language is a very common problem in live feeds on video streaming services (e.g., YouTube) (Liebeskind et al., 2021). Toxic comments can appear more frequently in live broadcasting, since users tend to impulsively post comments in real-time with less introspection during live streaming (Gao et al., 2020). It is not possible to manually rule out toxic comments from a large number of comments continuously posted across multiple live feeds. We aim for an automatic detection system to address toxic comments, such as Figure 1.

To capture powerful latent features for detecting toxic language, recent existing methods rely on

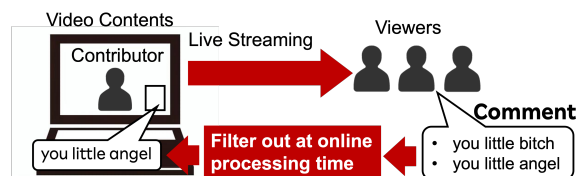


Figure 1: Diagram of Toxic Comment Detection System

deep learning techniques such as BERT. Although the techniques have performed well on the task, the following issues can be raised when assuming application to comments on live streaming.

1. Inference Speed: During live streaming, many posted comments can stream to all viewers in real-time. To promptly prevent both viewers and video contributors from reading toxic comments, online processing should be crucial. However, deep learning-based methods might not be scalable for online processing due to slow inference speed.

2. Computation Resource: A computational resource-friendly method is essential from an industry point of view. Even if the inference speed is satisfied with one or more GPUs at the PoC phase, there should be computational resource constraints, depending on the requirements of a deployed system when assuming real-world operations, such as API on a mobile application.

To make up for the above two issues, we propose an efficient method for toxic comment detection on live streaming chat. Our target includes offensive, insulting, and obscene expressions, similar to the expressions used by Leite et al. (2020). We handle Japanese comments since our method is meant to be deployed on a Japanese live streaming service. Our proposed architecture is a stacking-based two-layer classification model in which detection results with lower confidence scores in fastText classification are re-classified by LightGBM with five features. Thus, our method does not require a GPU environment. That enables a developer to facilitate deploying a system with low computation

resources and enough reproducibility. Experimental results show that our method achieves up to 17 times faster inference speed than several BERT-based methods with a comparable F1 score under an online processing setting.

2 Related Work

Many methods for toxic comment detection have been proposed, mainly for Twitter (Leite et al., 2020; Fehn Unsvåg and Gambäck, 2018; An et al., 2021; ElSherief et al., 2021; Founta et al., 2019) and comments to online news articles (Jigsaw, 2019; Baldini et al., 2022). Traditional methods use lexical patterns based on offensive words (Hosseini et al., 2017; Gröndahl et al., 2018) and opinion words (Pouran Ben Veyseh et al., 2022). Since malicious viewers create various toxic words in diverse writing styles, this approach is not robust to variants of words in the lexicon. In addition, it is costly to regularly update the lexicon to keep a deployed toxic detection system accurate (Nejadgholi et al., 2022).

In recent years, many methods utilize deep learning-based methods such as BERT and LSTM, using sentiment information (Brassard-Gourdeau and Khoury, 2019; Zhou et al., 2021; Cao et al., 2020; Pouran Ben Veyseh et al., 2022), topic contents (Almerekhi et al., 2020; Bose et al., 2021), and context information such as text replies (Dahiya et al., 2021; Bhat et al., 2021) and attention-based context vectors (Chakrabarty et al., 2019). Baldini et al. (2022) explored how BERT-based models affect the relationship between performance and fairness for toxic comment detection. Here, fairness means equalized performance across various sensitive groups such as religion and race. However, none of the studies in this section explore performance that consider inference speed and computational resources for toxic comment detection for real-world applications.

Comment characteristics in video live streaming chats are fundamentally different from Twitter posts and news articles. There is no information on replies (i.e., parent-child relationship) in live streaming chats. Moreover, the comments are often short, which lacks context and topic information. According to Yousukkee and Wisitpongphan (2021), 63% of messages in live streaming chats on YouTube contained fewer words than the average word count of 8 with standard deviations of 7.

In video live streaming chat on Twitch¹, Gao et al. (2020) applied a fine-tuned RoBERTa model to toxic comment detection. We show the efficiency of our method by making a comparison with various BERT-based models.

Edge computing can be an applicable solution to address latency and scalability challenges for NLP services with deep learning (Chen and Ran, 2019; Han et al., 2020). There is a wide range of deployed candidates for edge computing architectures and deep learning models. The deployment should be carefully considered to accomplish system requirements. Thus, we leave the applicability of deep learning with edge-computing in our task for future research.

3 Data Collection

We create annotated data for toxic comment detection on the video live streaming domain. We use “NicoNico-Doga comment data” (DWANGO Co., 2021-12-22) provided by the National Institute of Informatics². NicoNico-Doga is one of the largest-scale Japanese video streaming services. In the whole dataset, we used the file lists from 0000.zip to 0005.zip, and labeled them as toxic/non-toxic comments via human annotation. In total, 168,071 comments were annotated, which comprise 21,156 toxic comments and 146,915 non-toxic comments. We randomly divided the annotated dataset into a training set, development set, and test set at a ratio of 80%, 10%, and 10%, respectively. The statistics are shown in Table 1. To evaluate inter-annotator agreement, additional two Japanese annotators independently identified a toxic or non-toxic label to 2,122 comments from scratch. The inter-annotator agreement was $\kappa = 0.77$, which indicated substantial agreement. Figure 2 shows the distribution of comments divided by word count in our dataset. The distribution of our dataset is similar to one reported in Yousukkee and Wisitpongphan (2021) on YouTube live stream. In our dataset, 67% of comments contained fewer words than the average word count of 7 with standard deviations of 6.

4 Proposed method

4.1 Overview

Our task is to classify posted comments as toxic or non-toxic. As mentioned in §1, we assumed that

¹<https://www.twitch.tv>

²<https://www.nii.ac.jp/dsc/idr/nico/nicocomm-apply.html>

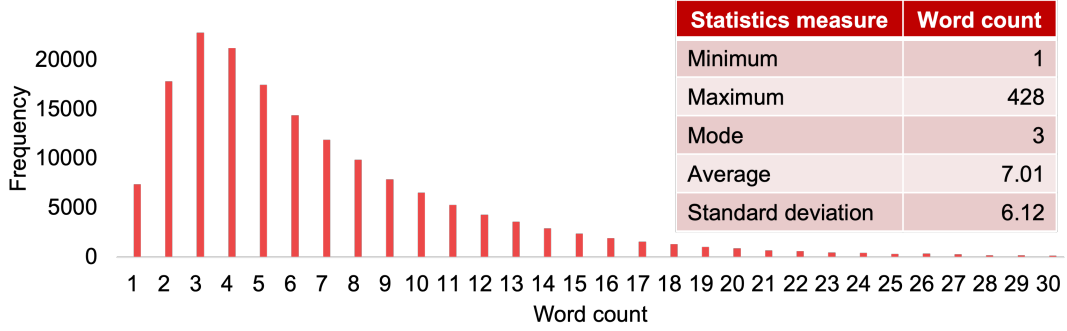


Figure 2: Number of comments divided by word count in our dataset

	Toxic	Non-toxic	Total
Train	16,925	117,532	134,457
Dev	2,116	14,692	16,808
Test	2,115	14,691	16,806
Total	21,156	146,915	168,071

Table 1: Statistics of dataset

online processing should be crucial for comments posted continuously in live streaming, and computational resources are constrained depending on the specification of the deployed system. Thus, we design a model architecture under the following two limitations for a deployed system.

- Avoid using a GPU resource
- Avoid using a deep learning model

Figure 3 shows the architecture of our proposed method, which comprises two layers. We first use fastText classification model (Joulin et al., 2017) as described in §4.2 in order to prioritize fast inference speed. Our preliminary experiments with our dataset showed that the lower the prediction probability for the fastText classification, the lower the F1 score. Thus, there is much room for improvement in results of lower prediction probability as described in the triangle area in Figure 4. The results motivate us to utilize another classification model for the case where the fastText model has less confidence.

After getting classification results by the fastText, we use the stacking technique (Džeroski and Ženko, 2004). This technique is a simple but effective way that predictions of different classifiers are fed to a meta-level classifier to generate final results. For efficiency, the second classifier is applied only to the first results with lower confidence. We

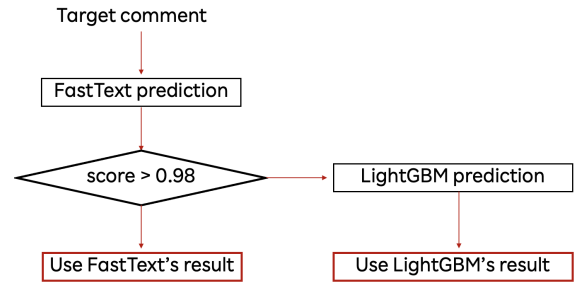


Figure 3: Model Architecture

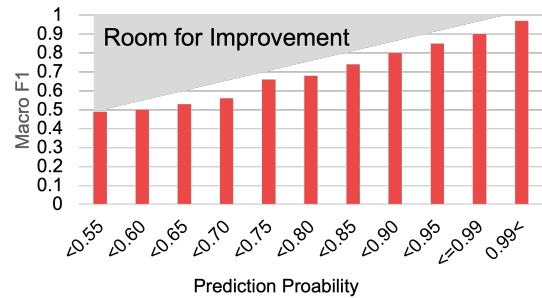


Figure 4: Macro F1-score by prediction probability for fastText classification model

will describe the meta classifier and used features in §4.3.

4.2 First layer: fastText classification model

The model is a simple neural network with only one layer. The bag-of-words representation is first fed into a lookup layer to obtain a word representation for every word. The model takes an average of word representations into a text representation, which is in turn fed to a linear classifier. We use the softmax function to compute the probability distribution over the two classes. We calculate confidence score of classification result by Equation (1)

$$\text{Conf}(c) = \max\{P(y = 1|c), P(y = 0|c)\} \quad (1)$$

Input comments are fed to the meta-level classifier if the confidence score is equal to or less than the threshold θ . Otherwise, we use fastText prediction as output.

4.3 Second layer: Meta-level Classification

As meta classifier, we select a LightGBM (Ke et al., 2017), gradient boosting decision tree-based model. The advantage of using LightGBM is its efficiency and interpretability. Error analysis is indispensable to keep a deployed system performance accurate for future data. LightGBM enables us to facilitate detecting a cause of an error through tracing decision flow. For the model training, we propose five features from three perspectives.

Two Lexicon features: #Black words and #Gray words Unlike existing studies, we make a lexical feature considering certainty for toxic words. The more a comment includes toxic words, the more the text is likely to be toxic. However, whether or not a word is associated with being toxic depends on context. For example, the Japanese word “くそ (sh*t)” has two word meanings, which are “very” and a literal toxic word. If a comment uses the word with the former meaning, a system can incorrectly identify the comment as a toxic comment. To alleviate this problem, toxic words are divided into two categories in terms of certainty, called “black words” and “gray words” in this paper. Black words are words considered toxic regardless of the context. Gray words are words that are not considered toxic based on the context. Based on those ideas, we use the number of black words and the number of gray words in a comment as feature values. We manually created 1,338 black words and 1,614 gray words.

fastText Prediction We use the prediction label as a feature value. Our preliminary experiments show that fastText classification model tends to return a low confidence score when ground-truth is a “toxic” label, as illustrated in Figure 5. To utilize this empirical finding, we also use the confidence score $\in (0.5, 1]$ computed in §4.2.

SVM Prediction As a third perspective, we use prediction results of Support Vector Machine (SVM) trained with TF-IDF weighting scheme. For each word w in a comment c , TF-IDF is calculated

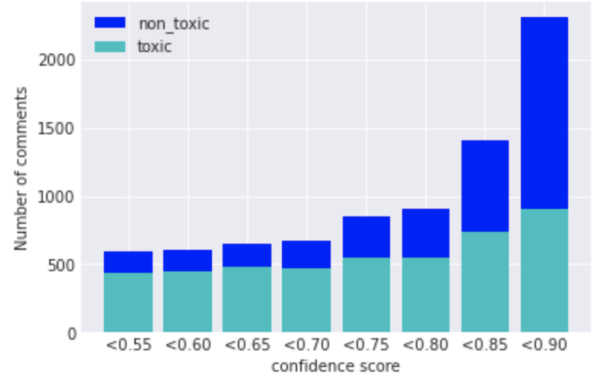


Figure 5: Ratio of toxic comments by confidence score

by the Equation (2).

$$\text{TF-IDF}(w, c) = \frac{f_{w,c}}{\sum_{w' \in c} f_{w',c}} \cdot \log \frac{N}{\text{df}(w) + 1} \quad (2)$$

where $f_{w,c}$ denotes frequency of w in c . $\text{df}(w)$ denotes the number of documents in which w appears. N is the total number of comments.

5 Experiments

5.1 Settings

We did experiments with the annotated dataset in §3 to show the effectiveness and the efficiency of our method. Model trainings were conducted on Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz with a single processor. RAM size is 26GB. Cache memory consists of 32KB for L1d, 32KB for L1i, 1024KB for L2, and 28,160KB for L3. The model implementation is as follows:

fastText classification To get the word representations, we create a pre-trained model using the fastText module (Bojanowski et al., 2017) with all the comments in §3. For pre-processing, this data is tokenized by the Japanese morphological analyzer MeCab (Kudo, 2006). Additionally, a word was lemmatized and half-width characters were converted to full-width. Hyperparameters for the pre-training are as follows: The number of dimensions for word representation is 300. We used skip-gram to train word representation. The threshold θ in §4.2 was determined with development set ($\theta = 0.98$).

SVM We calibrated the prediction results using a calibrated ClassifierCV³ provided by scikit-learn

³<https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html>

to remove the effect of bias in the unbalanced data.

LightGBM This model was optimized with optuna (Akiba et al., 2019).

As baseline models, we used each component of our method and existing pre-trained BERT-based models. We used the BERT-base⁴ and BERT-large⁵ models pre-trained from Japanese Wikipedia. Additionally, it is essential to compare with light BERT models for a fair evaluation of inference speed. Over the last couple of years, variants of BERT have been proposed to make the model size light and inference speed efficient. Specifically, we tried Distil-BERT⁶ (Sanh et al., 2019), ALBERT⁷ (Lan et al., 2019), and Poor-Man’s BERT (Sajjad et al., 2020). Poor-Man’s BERT is the method that removes some layers from an original BERT. Their experimental results showed that dropping the top layers works consistently well across different tasks when dropping 4 and 6 layers. Following those findings, we removed the top 4 layers in the BERT-base that we used.

For fine-tuning and testing for BERT-based methods, we used a single 32GB NVIDIA-V100 GPU. The batch size for fine-tuning was 16. We set batch size for inference to 1, since we assumed online processing at the inference phase. Another possible way of improving inference speed would be adjusting the maximum length of an input sequence. We explored the relationship between performance and inference speed on variation of the length (4, 8, 16, 32, and 64). We evaluated the average inference time over 10 trials.

5.2 Results

Table 2 shows classification results of our method obtained with the threshold optimized in terms of Macro F1 score. Our method achieved 0.942 in terms of average of Macro F1 across the two classes. Table 3 shows results and the effectiveness of each component. Looking at Table 3, one can see that our stacking-based method outperformed fastText single classification model by 3.3 points and the remaining ones as well. We observed that 24% of test samples proceeded to the meta-level

⁴<https://huggingface.co/cl-tohoku/bert-base-japanese-whole-word-masking>

⁵<https://huggingface.co/cl-tohoku/bert-large-japanese>

⁶<https://huggingface.co/bandainamco-mirai/distilbert-base-japanese>

⁷<https://huggingface.co/ALINEAR/albert-japanese-v2>

	Precision	Recall	Macro F1
Non toxic	0.981	0.991	0.986
Toxic	0.931	0.867	0.898
Macro Avg.	0.956	0.929	0.942

Table 2: Classification performance

Method	Macro F1
fastText classification	0.919
Black words	0.905
Gray words	0.860
SVM	0.905
Our method	0.942
w/o SVM prediction	0.940
w/o graywords	0.938
w/o fastText probability	0.937
w/o blackwords	0.933
w/o fastText prediction	0.932

Table 3: Performance Comparison and Ablation Study

classifier. In Table 3, our ablation study showed that our five features contributed to enhancing F1 score, especially black words and fastText prediction. Thus, we believe that each of our proposed features was independently effective for toxic language detection tasks, and that the improvement was even greater when used together.

Table 4 shows the comparison between our method and BERT-based methods on various configurations. The average inference time for our method was 22.9 seconds for 16,807 test samples, with a standard deviation of 1.18. In the Table, we put QPS score (i.e., Throughput), the number of comments which can be processed per a second. Looking at the table, our method achieved much faster inference speed (734QPS±41) than any other BERT configurations. We found that the differences in inference speed between our method and BERT models were statistically significant at the 1% level, irrespective of the configurations by the two-tailed paired t-test for statistical testing.

BERT-large and ALBERT yielded slightly better performance than our method when the maximum sequence length was 64 (F1 = 0.948) and 32 (F1 = 0.943), respectively. However, these models sacrificed inference speed. For instance, the inference speed (47QPS±6) of BERT-large is 1.7 times slower than BERT-base (82QPS±4) and 15.6 times much slower than our method. On the other hand, if attaching great importance to inference speed,

	F1	QPS	max_len = 4		max_len = 8		max_len = 16		max_len = 32		max_len = 64	
			F1	QPS	F1	QPS	F1	QPS	F1	QPS	F1	QPS
BERT-base			0.775	89±3	0.882	86±6	0.931	84±5	0.936	84±3	0.937	82±4
BERT-large			0.783	50±2	0.891	48±2	0.939	48±2	0.943	47±2	0.948	47±2
Distil-BERT			0.773	145±6	0.878	144±4	0.923	134±9	0.924	134±9	0.928	131±7
ALBERT			0.699	81±5	0.866	84±4	0.928	79±4	0.943	78±3	0.938	79±2
Poor Man’s BERT			0.774	114±7	0.882	115±9	0.930	112±4	0.936	112±6	0.935	109±6
Our method	0.942	734±41										

Table 4: Relationship between Macro-F1 and inference speed on various configurations

Distil-BERT achieved the highest inference speed (145QPS±6) of all the BERT-based settings when the maximum sequence length was 4, but F1-score dramatically went down to 0.773. It seems that Poor-Man’s BERT and Distil-BERT had harmonized results of BERT settings when the maximum length was 32 or 64. However, all of the values did not reach those for our method. Thus, we believe that our method is intended for real-world deployment in terms of low-cost computational resources with the comparable F1 score.

5.3 Error Analysis

To understand the difference between our method and the BERT-large, we analyzed error cases where the BERT-large made correct predictions whereas our method failed. We describe a cause of an error with the example comments in Table 5. In the Table, we input English translation from the original Japanese in parenthesis. On ethical grounds, some parts were replaced with “*”.

The total number of our target errors was 195, divided into 61 false positives and 134 false negatives. In the false positives, 39 cases (64%) contained our gray word. Thus, we suspected that the gray word affects classification results for some patterns. We observed two cases in which the gray word can be noise.

recognized sub-word In the comment (a) in Table 5, the word “きめつ” (kimetsu) is not a toxic word but just a title of Japanese animation. Our method mistakenly identified the sub-word “きめ (gross)” in the word as a gray word.

recognized without Word Sense Disambiguation

In the comment (b), the word “はげ” is often used as the toxic word “bald”. However, this word is sometimes also used as abbreviation of “はげしく (strongly)”. In this context, the word had the latter meaning and thus should not be identified as a gray word.

In the false negatives, 96 errors (72%) were classified as a non-toxic comment for both SVM and

fastText classification models despite the presence of gray words. Of the 96 errors, we identified that 39 cases (41%) would be due to either of the two causes.

Coarse-grained tokenization When the same word is written consecutively, such as the comment (c), the MeCab tokenizer did not split that phrase into a finer-grained word unit. We consider that our method could not capture a feature of being toxic due to a coarse-grained token with useless TF-IDF value and word representation.

lol (laugh out loud) slang expression SVM was trained so that the word “w (lol)” can contribute to being non-toxic, rather than toxic, since the word also appears in a non-toxic context. As a result, even though a gray word is included in a comment, the comment was not identified as a toxic comment if the expression is used many times in the comment (d).

Input	Gold	Ours
(a) だから、きめついらんよ (I told we don’t need kimetsu)	N	T
(b) はげど! (Strongly agreed!)	N	T
(c) エロエロエロエロ (EroticEroticErotic)	T	N
(d) タグまじかw*ねw w復帰スンナw (tag is serious?lol fu**k off and die lol lol don’t come back lol)	T	N

Table 5: Error cases (T: Toxic, N: Non-toxic)

6 Conclusion

Although many methods have been proposed to detect toxic comments on online social platforms, these methods have paid no attention to inference speed and constraints of computational resources for real-world applications. We presented a fast and computational resource-friendly method. Our method does not require GPU resources, which facilitate being adjustable with a requirement of a deployed system. We proposed a two-layer clas-

sification model that efficiently utilizes stacking techniques with five features. Experimental results showed that all of the proposed features were effective independently. Under the online processing setting, our method achieved a much faster inference speed than fine-tuned BERT-based methods, with the comparable F1 score. Our method is going to be deployed on our service soon. We leave the issues raised in error analysis for future research.

Acknowledgements

In this paper, we used "Nicovideo Comment etc. data" provided by DWANGO Co., Ltd. via IDR Dataset Service of National Institute of Informatics.

Ethical consideration

This study involves issues related to freedom of expression. Detecting and hiding inappropriate comments is an act that is closely associated with censorship. There is still room for detailed discussion on the extent to which it should be considered toxic.

References

- Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. [Optuna: A next-generation hyperparameter optimization framework](#).
- Hind Almerakhi, Supervised by Bernard J Jansen, and co-supervised by Haewoon Kwak. 2020. Investigating toxicity across multiple reddit communities, users, and moderators. In *Companion proceedings of the web conference 2020*, pages 294–298.
- Jisun An, Haewoon Kwak, Claire Seungeun Lee, Bongang Jun, and Yong-Yeol Ahn. 2021. [Predicting anti-Asian hateful users on Twitter during COVID-19](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4655–4666, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ioana Baldini, Dennis Wei, Karthikeyan Natesan Ramamurthy, Moninder Singh, and Mikhail Yurochkin. 2022. [Your fairness may vary: Pretrained language model fairness in toxic text classification](#). In *Findings of the Association for Computational Linguistics*, pages 2245–2262, Dublin, Ireland. Association for Computational Linguistics.
- Meghana Moorthy Bhat, Saghar Hosseini, Ahmed Hassan Awadallah, Paul Bennett, and Weisheng Li. 2021. [Say ‘YES’ to positivity: Detecting toxic language in workplace communications](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2017–2029, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. [Enriching word vectors with subword information](#). *Transactions of the Association for Computational Linguistics*, 5.
- Tulika Bose, Irina Illina, and Dominique Fohr. 2021. [Generalisability of topic models in cross-corpora abusive language detection](#). In *Proceedings of the Fourth Workshop on NLP for Internet Freedom: Censorship, Disinformation, and Propaganda*, pages 51–56, Online. Association for Computational Linguistics.
- 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.
- Rui Cao, Roy Ka-Wei Lee, and Tuan-Anh Hoang. 2020. Deepate: Hate speech detection via multi-faceted text representations. In *12th ACM conference on web science*, pages 11–20.
- 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, Florence, Italy. Association for Computational Linguistics.
- Jiasi Chen and Xukan Ran. 2019. [Deep learning with edge computing: A review](#). *Proceedings of the IEEE*, 107(8):1655–1674.
- Snehil Dahiya, Shalini Sharma, Dhruv Sahnan, Vasu Goel, Emilie Chouzenoux, Víctor Elvira, Angshul Majumdar, Anil Bandhakavi, and Tanmoy Chakraborty. 2021. Would your tweet invoke hate on the fly? forecasting hate intensity of reply threads on twitter. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2732–2742.
- Ltd. DWANGO Co. 2021-12-22. [Nicovideo comment etc. data](#). informatics research data repository, national institute of informatics (dataset).
- Saso Džeroski and Bernard Ženko. 2004. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273.
- Mai ElSherief, Caleb Ziems, David Muchlinski, Vaishnavi Anupindi, Jordyn Seybolt, Munmun De Choudhury, and Diyi Yang. 2021. [Latent hatred: A benchmark for understanding implicit hate speech](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 345–363, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Elise Fehn Unsvåg and Björn Gambäck. 2018. [The effects of user features on Twitter hate speech detection](#). In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 75–85, Brussels, Belgium. Association for Computational Linguistics.

- Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2019. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM conference on web science*, pages 105–114.
- Zhiwei Gao, Shuntaro Yada, Shoko Wakamiya, and Eiji Aramaki. 2020. [Offensive language detection on video live streaming chat](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1936–1940, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. [All you need is "love": Evading hate-speech detection](#). *CoRR*, abs/1808.09115.
- Yiwen Han, Xiaofei Wang, Victor C. M. Leung, Dusit Tao Niyato, Xueqiang Yan, and Xu Chen. 2020. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 22:869–904.
- Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google’s perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*.
- Kaggle Jigsaw. 2019. Jigsaw unintended bias in toxicity classification. [Online:Accessed: 2022-07-18].
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. [Bag of tricks for efficient text classification](#). In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. 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. *Advances in neural information processing systems*, 30.
- Taku Kudo. 2006. Mecab: Yet another part-of-speech and morphological analyzer. <http://mecab.sourceforge.jp>.
- 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](#).
- 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.
- Chaya Liebeskind, Shmuel Liebeskind, and Shoam Yechezkel. 2021. [An analysis of interaction and engagement in youtube live streaming chat](#). In *2021 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/IOP/SCI)*, pages 272–279.
- Isar Nejadgholi, Kathleen Fraser, and Svetlana Kiritchenko. 2022. [Improving generalizability in implicitly abusive language detection with concept activation vectors](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5517–5529, Dublin, Ireland. Association for Computational Linguistics.
- Amir Pouran Ben Veyseh, Ning Xu, Quan Tran, Varun Manjunatha, Franck Dernoncourt, and Thien Nguyen. 2022. [Transfer learning and prediction consistency for detecting offensive spans of text](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1630–1637, Dublin, Ireland. Association for Computational Linguistics.
- Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. 2020. [On the effect of dropping layers of pre-trained transformer models](#).
- Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).
- Sawita Yousukkee and Nawaporn Wisitpongphan. 2021. [Analysis of spammers’ behavior on a live streaming chat](#). *IAES International Journal of Artificial Intelligence (IJ-AI)*, 10:139–150.
- Xianbing Zhou, Yang Yong, Xiaochao Fan, Ge Ren, Yunfeng Song, Yufeng Diao, Liang Yang, and Hongfei Lin. 2021. [Hate speech detection based on sentiment knowledge sharing](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7158–7166, Online. Association for Computational Linguistics.

End-to-End Speech to Intent Prediction to improve E-commerce Customer Support Voicebot in Hindi and English

Abhinav Goyal, Anupam Singh, Nikesh Garera

Flipkart

{abhinav.goyal, anupam.s, nikesh.garera}@flipkart.com

Abstract

Automation of on-call customer support relies heavily on accurate and efficient speech-to-intent (S2I) systems. Building such systems using multi-component pipelines can pose various challenges because they require large annotated datasets, have higher latency, and have complex deployment. These pipelines are also prone to compounding errors. To overcome these challenges, we discuss an end-to-end (E2E) S2I model for customer support voicebot task in a bilingual setting. We show how we can solve E2E intent classification by leveraging a pre-trained automatic speech recognition (ASR) model with slight modification and fine-tuning on small annotated datasets. Experimental results show that our best E2E model outperforms a conventional pipeline by a relative $\sim 27\%$ on the F1 score.

1 Introduction

Spoken Language Understanding (SLU) systems that extract the intent from a spoken utterance are integral in various voicebot applications such as automated on-call customer support, voice assistants, home or vehicle automation systems, etc. The extracted intent triggers a standard operating procedure (SOP) as defined by the respective application, e.g. an e-commerce customer query “I want to return my phone” maps to “Return” intent which triggers the SOP to help the user with returns. It helps us reduce the reliance on human agents and provide faster resolutions. More elaborate examples are shown in Table 4.

Conventionally, such systems consist of two components - an Automatic Speech Recognition (ASR) system followed by a Natural Language Understanding (NLU) unit. ASR converts audio to text, and NLU performs intent classification. Further, each component can have multiple sub-models. Typically, both these components are developed and optimized independently. ASR optimizes word error rate (WER) with equal weightage

to individual words. This might not be optimal for an S2I system since all words are not equally relevant for intent classification. Also, due to the broad diversity in speech, training reliable ASR models can be very data intensive and strenuous. An error-prone ASR results in noisy inputs to NLU models, typically trained on clean text. This causes error accumulation which reduces the pipeline’s performance. Data-intensive training of multiple models, high complexity & maintenance, and higher overall latency make this pipeline approach sub-optimal.

The end-to-end S2I model is an intuitive alternative to overcome these limitations. It eliminates the problem of error accumulation, is simple and faster, and reduces the efforts required for independent models. Modelling the problem as audio-to-intent classification simplifies the task since the number of intents is usually much less than the vocabulary size used in ASR and NLU. It helps us reduce the requirement of manually annotated training data.

In this work, we adapt an E2E ASR model to build an E2E S2I model for Flipkart’s on-call customer support. An overview of our contributions is as follows:

- An efficient extension of end-to-end BiLSTM and CTC based ASR models for S2I task on noisy datasets;
- A demonstration of how the idea can outperform conventional pipeline in customer support voicebot in real-world settings;
- An investigation on how ASR pre-training, offline active learning and pseudo labelling reduce data labeling requirements for S2I.

Next, we discuss some related work in Section 2. Section 3 & 4 describe the baseline S2I pipeline and our E2E approach respectively. We talk about datasets, preprocessing and experimental setup in Section 5. Finally, we conclude with a discussion on results and limitations in Section 6.

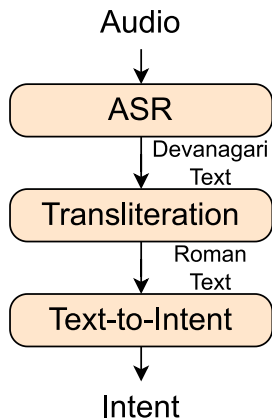


Figure 1: Text-based baseline system

2 Related Work

There have been several attempts to mitigate ASR error propagation in text-based pipelines. One straightforward idea is to correct the ASR output, using error correction models (Weng et al., 2020; Tam et al., 2014) or by ranking n-best hypotheses (Ogawa et al., 2018, 2019; Fohr and Ilina, 2021). Other approach is to leverage extra information from ASR - output lattice (Ladhak et al., 2016; Huang and Chen, 2019, 2020), n-best hypotheses (Morbini et al., 2012; Li et al., 2020; Liu et al., 2021) or word confusion networks/embeddings (Tür et al., 2002; Shivakumar et al., 2019). Though these approaches make NLU robust to some ASR errors, they still use a strict multi-component pipeline.

There have been an increasing number of attempts toward building end-to-end SLU models. Qian et al. (2017); Serdyuk et al. (2018); Chen et al. (2018) investigate end-to-end SLU models which do not use ASR at all whereas Haghani et al. (2018) optimizes ASR and NLU in a joint setup. Such end-to-end models can require a large amount of paired speech and intent data which may not always be available. Wang et al. (2020); Morais et al. (2021) explore unsupervised pre-training which helps in low-resource settings but is usually very compute intensive. An alternative approach is to initialize SLU models using weights trained for ASR (Lugosch et al., 2019; Kuo et al., 2020; Qian et al., 2021). Since ASR datasets are more easily available, this approach presents a much easier method of pre-training than unsupervised methods.

Inspired by ASR pre-training, we explore how to augment a pre-trained ASR model for end-to-end S2I task for Flipkart customer support voicebot in Hindi and English languages.

3 Text-based Pipeline

Our baseline consists of 3 components - ASR, transliteration and text-to-intent as shown in Fig. 1. We use a bilingual ASR system which predicts text in Devanagari script for both Hindi and English. The transliteration model converts this text into Roman script. Finally, the text-to-intent model extracts intent from Roman text.

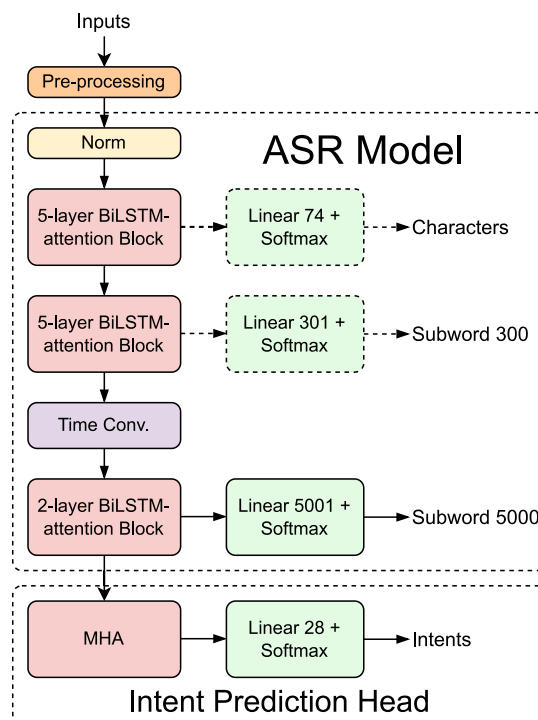


Figure 2: E2E Speech-to-Intent. Features from the last ASR block are used as inputs for intent classification.

Automatic Speech Recognition

Inspired by Fernández et al. (2007), we use a 3-level HCTC architecture based on LSTM and attention (Vaswani et al., 2017) as shown in Fig. 2. Going in a fine-to-course fashion, the model predicts characters (73 tokens), short subwords (300 tokens) and long subwords (5000 tokens) at the respective levels. We use unigram models from Sentencepiece (Kudo and Richardson, 2018) for text segmentation. Each level consists of an N-layer LSTM-attention block (Fig. 3), N being 5-5-2, followed by a linear softmax layer.

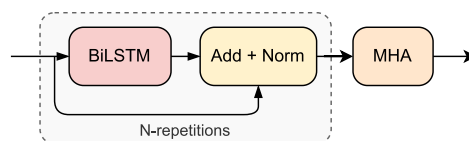


Figure 3: N-layer BiLSTM-attention block

For inference, the output of the last block, with 5000 subword units, is used for decoding the text using prefix beam search. The top 100 candidates are then re-ranked using a 3-gram KenLM (Heafield, 2011) to select the best one.

Transliteration

A manually curated mapping and a fallback transformer encoder-decoder model (Vaswani et al., 2017), with a single layer each in encoder and decoder, is used for transliteration. The transformer uses a sum of character and position embeddings as inputs. Together, this combination has a WER of <1% on unique utterances from a blind test set.

Text-to-Intent

For text-to-intent classification, we have 28 categories (26 intents + others + blank) related to different customer queries, e.g. "Delivery status", "Product return" etc. We use the "Blank" intent when the output text is blank. For the baseline, we try different models, out of which XGBoost (Chen and Guestrin, 2016) with TF-IDF features gives the best results. We observe that neural network-based models - BiLSTM and BERT (Devlin et al., 2018) overfit on our dataset. BERT when pre-trained on a large corpus performs at par with XGBoost.

4 E2E Speech-to-Intent

For the S2I task, we augment the pre-trained ASR model (same as used in the baseline) with intent prediction head as shown in Fig. 2. We summarize the hidden features from the last block of the ASR model using a dot-product based multi-headed self-attention (MHA) layer. We use the output sequence of the last block as key-value vectors and the final cell state of the last BiLSTM layer as the query vector. A linear layer then predicts probability distribution over the intent classes. Since there's no text output from the model, the "Blank" intent is also predicted the by E2E S2I model. We train the intent prediction head (and fine-tune the BiLSTM blocks) using cross entropy loss.

5 Experiments

5.1 Datasets

Automatic Speech Recognition

A collection of datasets is used to train the ASR model - Flipkart customer support voicebot queries, voice search queries and general domain speech

data. We transcribe all the utterances using an existing ASR system and manually correct the errors. The ASR system used to generate reference text is incrementally improved as more data is available. There's no control over the recording environment, and the correction of ASR transcripts instead of transcription from scratch leaves some errors introduced by the ASR model. This causes the dataset to have a lot of acoustic and textual noise. The datasets collectively amount to ~11 M audio-text pairs which correspond to roughly 17k hours of audio. It has a mix of Hindi and English (possibly code-mixed) languages.

We train KenLM and Sentencepiece on a large corpus collected from various sources such as Flipkart's customer support chatbot and voicebot, voice search queries and product catalogue. The ~920k Voicebot utterances (in-domain data) are upsampled during training.

Transliteration

The transformer model is trained on ~96k unique words which are manually transliterated. This dataset consists of high frequency words in Hindi and English in equal proportions. We add manual transliterations of words frequent for our use case in the look-up dictionary.

Text-to-Intent

For training text-to-intent, ~90k manually labeled unique text-intent pairs are used. This mainly consists of customer support voicebot and chatbot queries. For deep neural-network based models - BiLSTM and BERT, a large pre-training corpus from e-commerce domain is also used.

Speech-to-Intent

For fine-tuning the model for S2I, we use a set of 10k randomly sampled voicebot queries (we call it V1) and manually label the intents. We also use additional 25k audios for offline active learning. We name this complete 10k+25k set as V2.

Since the legacy system uses independent models, training data, to be annotated, was sampled independently and randomly for each model. The training datasets for ASR and Text-to-Intent don't have a large intersection. Therefore, we don't have a large dataset for training E2E S2I models and instead use smaller, independently labelled datasets.

5.2 Pre-processing and Experimental Setup

We use standard log-mel-spectrogram features with a window of 20ms, a stride of 10ms, and FFT size

Task	Model	Source	Rough Size of Dataset
ASR	BiLSTM	Voicebot	893 hrs
		Voice Search	9.9k hrs
		Generic	6.4k hrs
	SPM, KenLM	Voicebot	920k sentences
		Others	10M sentences
Transliteration	Transformer	Generic	96k word pairs
Text-to-Intent	all	Voicebot	55k text-intent pairs
		Chatbot	35k text-intent pairs
Speech-to-Intent	V1	Voicebot	10k audio-intent pairs
	V2	Voicebot	10k+25k audio-intent pairs

Table 1: Breakup of various datasets used for training.

of 512. The number of filterbanks used is 80. We use masking (Park et al., 2019) for data augmentation. We also stack 5 consecutive frames with a stride of 3 frames giving an input feature vector of 400 size with a receptive field of 60ms and stride of 30ms for each time step.

We use a cyclical learning rate (LR) (Smith, 2017) to train the ASR model for 8 epochs with a batch size of ~ 42 minutes. For S2I, we use constant LR, batch size of ~ 26 minutes and fine-tune it in 2 steps - on V1 dataset for 10 epochs + V2 dataset for 6 epochs. Training takes ~ 2.5 days for ASR and ~ 24 minutes for S2I on 1 A100 GPU.

6 Results

We compare the baseline and E2E model on 14606 voicebot queries manually transcribed and annotated for text and target intent resp. We report accuracy and F1 score for intent classification and word error rate (WER) for ASR in Table 2. The ASR system used for baseline has a WER of 8.34%. As mentioned earlier in Section 3, transliteration module has a WER of $<1\%$. Together, the WER of the ASR + transliteration system becomes $<9.2\%$. The text-to-intent model has an F1 score of 85.84%. We compute this using manual transcriptions as inputs to the text-to-intent model.

The S2I model, fine-tuned on just 10k manually annotated audio-intent pairs (V1), outperforms the baseline by an absolute 3.07% on the F1 score. Using this, we predict intent on an unlabeled set and get a random sample of 25k audios where the model has low confidence (prediction probability as given by softmax on the last layer). We correct this set manually and re-train the model using all 35k samples (V2), improving the F1 score by 1.28%. We then re-train the model on the complete set of

voicebot queries (~ 920 k audios from ASR dataset) using pseudo labels, further improving the score marginally. Our final E2E model outperforms the baseline by an absolute 4.59% on the F1 score.

The E2E model has a median latency of ~ 41 ms, which is 1/3rd of the baseline latency (~ 123 ms). Since we can deploy the complete model on a GPU, it can handle inference at a much larger scale than the baseline - more than 1000 queries per second using a single A100 GPU. Whereas the decoder in the ASR system used for the baseline, which is the bottleneck, can only handle about 90 queries per second. Thus, the E2E model outperforms the baseline on accuracy, latency, and scalability.

6.1 Analysis and Discussion

We also evaluate a simple time average of sequence output from the last ASR block in place of MHA. It gives almost the same results as MHA showing that the ASR model can adapt to intent classification task without extra modelling efforts. We observe that training the S2I model from scratch performs very suboptimal, which shows the importance of initializing the network using the ASR task when paired audio-intent data is scarce.

The text-to-intent model has a higher F1 score than the complete pipeline (85.84% vs 82.78%), suggesting that errors by the ASR model are the reason for the baseline’s suboptimal performance. Our S2I model is not only able to mitigate this but also gives more improvement as it is 1.53% better than standalone text-to-intent with manual transcriptions. Table 3 shows some examples to compare our S2I model with the baseline. In examples 1-3, ASR makes a mistake due to wrong pronunciation in 1 and high background noise in 2 & 3. These errors cause the text-to-intent model

Model	#params (in M)	WER	All Intents		Except blank/other	
			Accuracy	F1 score	Accuracy	F1 score
Baseline	48.60	8.34	83.62	82.78	82.76	85.36
Baseline with GT text	-	0	86.22	85.84	82.01	84.76
S2I linear (V1)	43.85	-	86.22	85.85	84.35	87.08
S2I MHA (V1)	45.97	-	86.28	85.85	85.88	87.81
S2I MHA from scratch (V1)	45.97	-	70.06	69.31	60.69	61.17
S2I MHA (V2)	45.97	-	87.18	87.13	87.60	89.94
S2I MHA (V2+pseudo lab.)	45.97	-	87.49	87.37	87.00	89.57

Table 2: Results on Intent Prediction. “Baseline” is the text based pipeline where text is given by the ASR system. “Baseline with GT text” is where we substitute ASR with true transcriptions. All numbers are in %.

#	Utterance	ASR output	True intent	Baseline	Ours
1	Wapas karne ka hai	Wapis karne ka hai	Return	Others	Return
2	Das <i>din</i> ke ander mujhe delivery chahiye	Das <i>June</i> ke ander mujhe delivery chahiye	Specific delivery time	Delivery info	Specific delivery time
3	Meri <i>watch</i> khrab hai	Meri <i>bahut</i> khrab hai	Return	Others	Return
4	Ji zaroor kariye	Ji zaroor kariye	Yes	Others	Yes
5	Delivery <i>ki</i> timing	Delivery <i>ke</i> timing	Delivery time	Delivery time	Delivery info
6	Kuchh nhi haan haan	Kuchh nhi haan haan	End	End	Yes

Table 3: Speech-to-Intent examples. In 1-4, our model does better and in 5 & 6, baseline does better.

to give wrong predictions demonstrating how error propagation affects the pipeline. In example 4, the intent model makes an error even with the correct transcription. In examples 5 and 6, the baseline outputs the right intent but the E2E model makes mistakes. In both cases, the E2E model confuses the intent with another very close category.

6.2 Conclusions

In this work, we show that pre-trained CTC-based end-to-end ASR models can be adapted for end-to-end Speech-to-Intent classification with slight augmentation and relatively much less annotated data. Our S2I model outperforms the text-based pipeline by an absolute 3.07% on the F1 score while keeping the model size small and requiring only 10k annotated audio-intent pairs to train. It also simplifies the pipeline by eliminating the requirement of a dedicated ASR decoder, Text-to-Intent model, and language models. With just 25k additional labelled training pairs, our final model is $\sim 27\%$ better than the baseline on the F1 score (absolute improvement of 4.59%). Thus, we show that the E2E S2I model, adapted from ASR, outperforms the conventional pipeline on accuracy, latency, and scalability while requiring much less labelled training data, compute resources, and modelling efforts.

Limitations

The baseline text-to-intent model was trained on a different dataset from what is used for fine-tuning the E2E models. But, using a considerably smaller dataset than the baseline system puts the E2E models at a disadvantage. Since the E2E models still outperform the baseline, the absolute difference in utterances should not affect the credibility of the results. We also need a transliteration module in the text-based pipeline because of a mismatch in scripts used by the ASR and Text-to-Intent models. One might argue that the transliteration module weakens the baseline. But being a highly accurate system (with $<1\%$ WER), it should not affect the baseline much. We have also verified this with a manual analysis of errors. Also, the ASR pre-training uses a large dataset which might not be readily available for low-resource languages.

As the future steps, it would be interesting to see how such models would work with more than two languages, where some of the languages are low-resource from ASR perspective. In such a setting, the ASR models usually employ a larger vocabulary to accommodate all languages, but since intent categories are the same, adapted S2I models can reduce vocabulary size by a greater extent giving them an even bigger advantage.

#	Customer Utterance	Intent/SOP	Voicebot Response
		Welcome	Namaste, aapka svagat hai. Main aapki automated support assistant. <i>(English: Hello and welcome. I am your automated support assistant)</i>
1		Order identification	Kya aap apne “curtain” ke bare mein baat karna chahte hain, jo “shipped” hai? <i>(English: Do you want to ask about your “curtain”, which is “shipped”)</i>
	Haan <i>(English: Yes)</i>	Yes/Confirmation	Aapka order ship ho chuka hai, aur aapko aaj mil jayega. Aapko kya madad chahiye? <i>(English: Your order has been shipped and you will get it by today. What do you need help with?)</i>
	Mera order aaj kitne time tak ghar pe aa jayega <i>(English: By what time today will my order reach?)</i>	Delivery Info	Hamare agent aaj shaam 6 bje tak deliver karne ki koshish karenge. Yadi nahi ho paya to agle karyadin deliver karenge. <i>(English: Our agents will try to deliver it by 6 pm today. If not possible, they will deliver it the next working day.)</i>
2		Order identification	Kya aap “usha air cooler” ke bare poochhna chahte hain, jo “returned” hai? <i>(English: Are you calling about “Usha Air Cooler”, which is “returned”)</i>
	Yes	Yes/confirmation	What do you need help with?
	When will I get my replaced one cooler?	Return status	Your replacement will be delivered within 2 days after pickup of returned product.
3		Order identification	Aapne haal hi mein ek se zyada order place kiye hain. Aaj apne kis order ke bare mein call kiya hai? <i>(English: Recently you placed more than 1 order. Which order are you calling about?)</i>
	Integriti jeans		Apka “integriti jeans” ka order delivered hai. Aapko is order ke bare mein kya madad chahiye? <i>(English: Your order of “integriti jeans” has been delivered. What help do you need with this order?)</i>
	Customer executive se baat karwao. <i>(English: Let me talk to a customer executive.)</i>	Agent request	Filhal hamare customer support executive vyast hain, aapse subah 9:51 sampark karenge. Kya main call ki vyavastha kar du? <i>(English: Currently our executives are busy. They could call you at 9:51 am in the morning. Should I arrange for a call back?)</i>
	Haan thik hai. <i>(English: yes it's fine.)</i>	Yes/Confirmation	Thik hai, maine call ki vyavastha kar di hai. <i>(English: Ok, I have arranged for a call back.)</i>
4		Order identification	Kya aapne TV ke bare mein call kiya hai? <i>(English: Have you called about TV?)</i>
	Haan ji <i>(English: Yes)</i>	Yes/Confirmation	Aapka order ship ho chuka hai aur 19 November tak deliver ho jaega. Aapko iss bare mein kya madad chahiye? <i>(English: Your order has been shipped and will be delivered by 19th November. What help do you need regarding this?)</i>
	Please deliver it today itself	Faster delivery request	Sorry, while faster delivery is not available, please be assured we will deliver your order on time.
		Conversation end	*** mein call karne ke liye dhanyavaad, apka din shubh rahe. <i>(English: Thanks you for calling ***, have a good day.)</i>

Table 4: Illustrations of how an S2I system can help provide faster and automated resolutions in e-commerce.

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*, page 785–794, New York, NY, USA. Association for Computing Machinery.
- Yuan-Ping Chen, Ryan Price, and Srinivas Bangalore. 2018. [Spoken language understanding without speech recognition](#). In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6189–6193.
- 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*.
- Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. 2007. [Sequence labelling in structured domains with hierarchical recurrent neural networks](#). In *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*.
- Dominique Fohr and Irina Illina. 2021. [Bert-based semantic model for rescoring n-best speech recognition list](#). In *INTERSPEECH 2021*.
- Parisa Haghani, Arun Narayanan, Michiel Bacchiani, Galen Chuang, Neeraj Gaur, Pedro Moreno, Rohit Prabhavalkar, Zhongdi Qu, and Austin Waters. 2018. [From audio to semantics: Approaches to end-to-end spoken language understanding](#). In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 720–726.
- 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.
- Chao-Wei Huang and Yun-Nung Chen. 2019. [Adapting pretrained transformer to lattices for spoken language understanding](#). In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 845–852.
- Chao-Wei Huang and Yun-Nung Chen. 2020. [Learning spoken language representations with neural lattice language modeling](#). *arXiv preprint arXiv:2007.02629*.
- 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.
- Hong-Kwang J Kuo, Zoltán Tüske, Samuel Thomas, Yinghui Huang, Kartik Audhkhasi, Brian Kingsbury, Gakuto Kurata, Zvi Kons, Ron Hoory, and Luis Lastras. 2020. [End-to-end spoken language understanding without full transcripts](#). *arXiv preprint arXiv:2009.14386*.
- Faisal Ladhak, Ankur Gandhe, Markus Dreyer, Lambert Mathias, Ariya Rastrow, and Björn Hoffmeister. 2016. [Latticernn: Recurrent neural networks over lattices](#). In *Interspeech*, pages 695–699.
- Mingda Li, Weitong Ruan, Xinyue Liu, Luca Soldaini, Wael Hamza, and Chengwei Su. 2020. [Improving spoken language understanding by exploiting asr n-best hypotheses](#). *arXiv preprint arXiv:2001.05284*.
- Xinyue Liu, Mingda Li, Luoxin Chen, Prashan Wani-gasekara, Weitong Ruan, Haidar Khan, Wael Hamza, and Chengwei Su. 2021. [Asr n-best fusion nets](#). In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7618–7622.
- Loren Lugosch, Mirco Ravanelli, Patrick Ignoto, Vikrant Singh Tomar, and Yoshua Bengio. 2019. [Speech model pre-training for end-to-end spoken language understanding](#). *arXiv preprint arXiv:1904.03670*.
- Edmilson Morais, Hong-Kwang J. Kuo, Samuel Thomas, Zoltán Tüske, and Brian Kingsbury. 2021. [End-to-end spoken language understanding using transformer networks and self-supervised pre-trained features](#). In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7483–7487.
- Fabrizio Morbini, Kartik Audhkhasi, Ron Artstein, Maarten Van Segbroeck, Kenji Sagae, Panayiotis Georgiou, David R. Traum, and Shri Narayanan. 2012. [A reranking approach for recognition and classification of speech input in conversational dialogue systems](#). In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 49–54.
- Atsunori Ogawa, Marc Delcroix, Shigeki Karita, and Tomohiro Nakatani. 2018. [Rescoring n-best speech recognition list based on one-on-one hypothesis comparison using encoder-classifier model](#). In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6099–6103.
- Atsunori Ogawa, Marc Delcroix, Shigeki Karita, and Tomohiro Nakatani. 2019. [Improved deep duel model for rescoring n-best speech recognition list using backward lstmlm and ensemble encoders](#). In *Interspeech*, pages 3900–3904.
- 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](#). *arXiv preprint arXiv:1904.08779*.
- Yao Qian, Ximo Bianv, Yu Shi, Naoyuki Kanda, Leo Shen, Zhen Xiao, and Michael Zeng. 2021. [Speech-language pre-training for end-to-end spoken language](#)

- understanding. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7458–7462.
- Yao Qian, Rutuja Ubale, Vikram Ramanaryanan, Patrick Lange, David Suendermann-Oeft, Keelan Evanini, and Eugene Tsuprun. 2017. [Exploring asr-free end-to-end modeling to improve spoken language understanding in a cloud-based dialog system](#). In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 569–576.
- Dmitriy Serdyuk, Yongqiang Wang, Christian Fuegen, Anuj Kumar, Baiyang Liu, and Yoshua Bengio. 2018. [Towards end-to-end spoken language understanding](#). In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5754–5758.
- Prashanth Gurunath Shivakumar, Mu Yang, and Panayiotis Georgiou. 2019. [Spoken language intent detection using confusion2vec](#). *arXiv preprint arXiv:1904.03576*.
- Leslie N. Smith. 2017. [Cyclical learning rates for training neural networks](#). In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472.
- Yik-Cheung Tam, Yun Lei, Jing Zheng, and Wen Wang. 2014. [Asr error detection using recurrent neural network language model and complementary asr](#). In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2312–2316.
- Gökhan Tür, Jerry H Wright, Allen L Gorin, Giuseppe Riccardi, and Dilek Hakkani-Tür. 2002. [Improving spoken language understanding using word confusion networks](#). In *Interspeech*.
- 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.
- Pengwei Wang, Liangchen Wei, Yong Cao, Jinghui Xie, and Zaiqing Nie. 2020. [Large-scale unsupervised pre-training for end-to-end spoken language understanding](#). In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7999–8003.
- Yue Weng, Sai Sumanth Miryala, Chandra Khatri, Runze Wang, Huaixiu Zheng, Piero Molino, Mahdi Namazifar, Alexandros Papangelis, Hugh Williams, Franziska Bell, and Gokhan Tur. 2020. [Joint contextual modeling for asr correction and language understanding](#). In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6349–6353.

PILE: Pairwise Iterative Logits Ensemble for Multi-Teacher Labeled Distillation

Lianshang Cai*, Linhao Zhang*, Dehong Ma†, Jun Fan
Daiting Shi, Yi Wu, Zhicong Cheng, Simiu Gu, Dawei Yin†

Baidu Inc., Beijing, China

{cailianshang, zhanglinhao, madehong, fanjun}@baidu.com
{shidaiting01, wuyi01, chengzhicong01, gusimiu}@baidu.com
yindawei@acm.org

Abstract

Pre-trained language models have become a crucial part of ranking systems and achieved very impressive effects recently. To maintain high performance while keeping efficient computations, knowledge distillation is widely used. In this paper, we focus on two key questions in knowledge distillation for ranking models: 1) how to ensemble knowledge from multi-teacher; 2) how to utilize the label information of data in the distillation process. We propose a unified algorithm called Pairwise Iterative Logits Ensemble (PILE) to tackle these two questions simultaneously. PILE ensembles multi-teacher logits supervised by label information in an iterative way and achieved competitive performance in both offline and online experiments. The proposed method has been deployed in a real-world commercial search system.

1 Introduction

Search engines have been an infrastructure in the Information Age to satisfy people’s needs for querying about information. In modern search engines, multi-stage pipelines are usually employed where *ranking* is usually known as the very final stage. It takes as input the shortlisted candidates of relevant documents (i.e. web pages) retrieved from previous stages, and concentrates on sorting (Lin et al., 2021) based on the degree of match between the latent semantics of documents and the search intent of the user’s query.

With the flourishing of pre-trained language models (Devlin et al., 2018; Sun et al., 2019b; Lan et al., 2019; He et al., 2020), BERT-based models have achieved state-of-the-art performance in a broad range of downstream tasks and text ranking is no exception. Pre-trained rankers based on BERT show impressive performance in ranking

tasks (Nogueira and Cho, 2019; Nogueira et al., 2019; Zou et al., 2021).

Despite the state-of-the-art performance pre-trained models yield in laboratories, it is hardly possible to apply them directly in real-world search engines. Their large numbers of parameters go against computational efficiency while the online environment is strictly restricted in resources. Therefore, before deploying a pre-trained model online, one necessary procedure is to reduce computational costs.

Knowledge distillation is one of the most commonly used methods to reduce the model size (Cristian et al., 2006; Hinton et al., 2015a) and accelerate the computation process. In a standard workflow of distillation, a large model (i.e. teacher) is pre-trained and finetuned well in advance, and a small model (i.e. student) imitates the teacher model’s behaviors. The knowledge learned by the teacher model is then transferred to the student model.

One of the risk factors hindering the improvement of the student model is that the knowledge acquired by a single teacher may be insufficient and biased. A straightforward solution is using an ensemble of multiple teachers for knowledge transfer. The ensemble process takes the predictions of multiple teachers into account and provides comprehensive guidance that helps to improve student performance (You et al., 2017). However, these teachers sometimes conflict with each other, and the heuristic of treating them equally by taking the mean of their predictions ignores the fact that they vary in confidence and correctness, thus often leading to suboptimal performance (Du et al., 2020). Besides, the valuable label information is ignored. Hence, how to ensemble knowledge from multi-teacher and how to utilize the label information are two key questions during the distillation process.

In this work, we introduce a unified algorithm called **Pairwise Iterative Logits Ensemble (PILE)** to tackle these two questions simultane-

*Equal contribution.

† Corresponding authors.

ously. The key idea of PILE is to assign a higher weight to teachers that produce more consistent soft targets with the golden labels. The resulting soft targets not only retain the generalization information transferred from the teacher models but also are integrated with the label information annotated by human experts.

We conduct both offline and online experiments and the results validate the effectiveness of PILE. The main contributions of this paper can be summarized as follows:

- We propose PILE, a specially designed ensemble algorithm to tackle multi-teacher distillation and labeled distillation. To the best of our knowledge, PILE is the first work that addresses these two key questions simultaneously in the ranking scenario.
- We conduct extensive offline and online experiments to demonstrate the effectiveness of our proposed method. The results show that PILE effectively boosts a real-world search engine’s performance.

2 Related Work

Text Ranking The goal of text ranking is to generate an ordered list of texts in response to a query. Conventional learning-to-rank (LTR) techniques (Li, 2014) are widely used for text ranking, which plays an important role in a wide range of applications, like search engines and recommender systems. LTR techniques can be roughly categorized into three types: pointwise approach (Cooper et al., 1992; Li et al., 2007), pairwise approach (Joachims, 2002; Zheng et al., 2007), and listwise approach (Cao et al., 2007; Burges, 2010). The former two are more widely used in practice as they are easier to optimize.

Recently, deep learning approaches have been widely adopted in ranking and BERT-based ranking models achieve state-of-the-art ranking effectiveness. For example, Nogueira and Cho (2019) use BERT-large (Devlin et al., 2018) as the backbone and feed the concatenation of query and passage text to estimate the relevant scores for passage re-ranking. Nogueira et al. (2019) formulate the ranking problem as a pointwise and pairwise classification problem and tackle them with two BERT models in a multi-stage ranking pipeline. Yilmaz et al. (2019) aggregate sentence-level information to estimate the relevance of the documents and

transfer the learned model to capture cross-domain notions of relevance.

However, the performance improvement comes at the cost of efficiency, which limits their real-world application. There are several ways to maintain high performance while keeping efficient computations for BERT-based models, such as knowledge distillation (Hinton et al., 2015b), weight sharing (Lan et al., 2019), pruning (Pasandi et al., 2020; Xu et al., 2020), and quantization (Hubara et al., 2017; Jacob et al., 2018). In this paper, we focus on knowledge distillation which has proven a promising way to compress large models while maintaining performance.

Knowledge Distillation The idea of knowledge distillation was first introduced by Cristian et al. (2006) to train small and fast models to mimic cumbersome and complex models, without much loss in performance. Hinton et al. (2015a) developed this idea further by minimizing the difference between their soft target distribution. With the rise of the pre-training and fine-tuning paradigm, various work has later extended this idea to large-scale pre-trained models and shown impressive results on multiple NLP tasks (Wang et al., 2019; Rajpurkar et al., 2018; Lai et al., 2017) with a significant gain in training efficiency. Sanh et al. (2019) conducted knowledge transfer during the pre-training phase, also known as a task-agnostic way. Sun et al. (2019a) proposed an approach to transfer knowledge between intermediate layers in the fine-tuning stage. Jiao et al. (2020) additionally uses attention-based distillation and hidden states-based distillation for students to imitate teachers’ behaviors in intermediate layers. Wang et al. (2020) introduced self-attention relation-based transfer and teacher assistants (Mirzadeh et al., 2020) to further improve the performance of students.

Ensemble Knowledge Distillation There is also some other work exploring the issues of multi-teacher distillation. For example, Du et al. (2020) adaptively ensemble knowledge distillation to find a better optimizing direction for the student network. Wu et al. (2021) designed a co-finetuning framework to jointly finetune multiple teachers for better collaborative knowledge distillation. Li et al. (2021) explored the influence of teacher model adoption which is promising for improving student performance. Different from the above work, we investigate the problem of ensemble knowledge distillation in ranking tasks and use the golden label

to supervise the ensemble process.

3 Methodology

3.1 Ranking Task Definition

In a search system, the ranking task aims to measure the relative order of a set of documents $D_q = \{d_i\}_{i=1}^N$ given a query $q \in Q$, where Q is a set of user queries and $D_q \subset \mathbb{D}$ is a set of q -related documents retrieved from a large document corpus \mathbb{D} (Liu et al., 2021). The ranking model determines the order of documents by computing the relevance score $f(q, d; \theta)$ of each query-document pair $\{(q, d_i)\}_{i=1}^N$, where f is a scoring function parameterized by θ representing the relevance of query q and document d .

As regards training procedure, the ranking model is learned by minimizing the empirical loss over the training data as

$$\mathcal{L} = \sum_{q \in Q} l(Y_{D_q}, F(q, D_q)),$$

where l is the loss function in learning to rank, e.g. pointwise loss, pairwise loss or listwise loss, and $F(q, D_q) = \{f(q, d_i)\}_{i=1}^N$ is a set of relevance scores, $Y_{D_q} = \{y_i\}_{i=1}^N$ is a set of labels. The label y_i is often assigned an integer range from 0 to 4, representing the relevance of the query-document pair (q, d_i) .

3.2 Knowledge Distillation

Due to the resource constraint, the ranking model can not directly serve online in a real production environment and we use knowledge distillation (KD) to compress the model size. In a commonly used KD framework, a large teacher model T is pre-trained or finetuned well in advance, and the knowledge of the teacher is transferred to a small student S by minimization of the difference between them, which can be formulated as:

$$\mathcal{L}_{KD} = \sum_{x \in \mathcal{D}} L(f^S(x), f^T(x)),$$

where \mathcal{D} denotes the training dataset and x is the input sample, $f^S(\cdot)$ and $f^T(\cdot)$ represent behavior measurements of teacher and student models, and $L(\cdot)$ is a loss function to measure the difference between their behaviors. The behaviors are usually represented by soft target distributions of the last layer, hidden state distributions or other deep semantic features such as self-attention distributions

and embedding layer outputs (Hinton et al., 2015a; Sun et al., 2020; Jiao et al., 2020; Wang et al., 2020, 2021). The methods that transfer the knowledge between the internal layers are limited in generality since the teachers and students are required to have the same model structure or align with each other in the number of layers or the size of hidden layers. Based on this consideration of generality, we perform knowledge distillation on the last layer only.

3.3 Pairwise Iterative Logits Ensemble

Knowledge distillation by one single teacher may bring some bias to the student model while simple yet common mitigation is distillation on the average of logits output by n multiple teachers. However, the teachers with diversity may conflict with each other since the biased teacher contributes equally as the unbiased teacher, which corrodes the confidence of distillation logits. Since the logits produced by teachers on different data vary in the degree of confidence, we conduct a dynamic weighting process for the ensemble. In the ranking task, the logit predicted by the teacher for a document represents a measurement of relevance with the query. The larger logits represent more correlation between the query and the document, and the correlation information is already annotated in its label. Thus, we consider utilizing the golden label to direct the assignment of weight. The procedures of the PILE algorithm are provided in Figure 1.

We start with initializing the ensembled distillation logit $e^{(0)}(q, d)$ for each query q and document d by way of averaging each teacher’s outputs:

$$\begin{aligned} w_k(q, d) &= 1 \\ e^{(0)}(q, d) &= \frac{1}{Z(q, d)} \sum_k w_k(q, d) f_k(q, d) \\ Z(q, d) &= \sum_k w_k(q, d), \end{aligned}$$

where $f_k(q, d)$ represents the relevance score predicted by k -th teacher and $w_k(q, d)$ is its weight w.r.t query q and document d .

Then, we perform the iterations of the update procedure. At iteration t , we randomly choose a pair of docs (d_i, d_j) related to the same query q , and check whether the magnitude of their ensemble logits is consistent with their labels. More specifically, if label $y_i > y_j$ and ensemble logits $e^{(t)}(q, d_i) < e^{(t)}(q, d_j)$, we call this pair of docs in

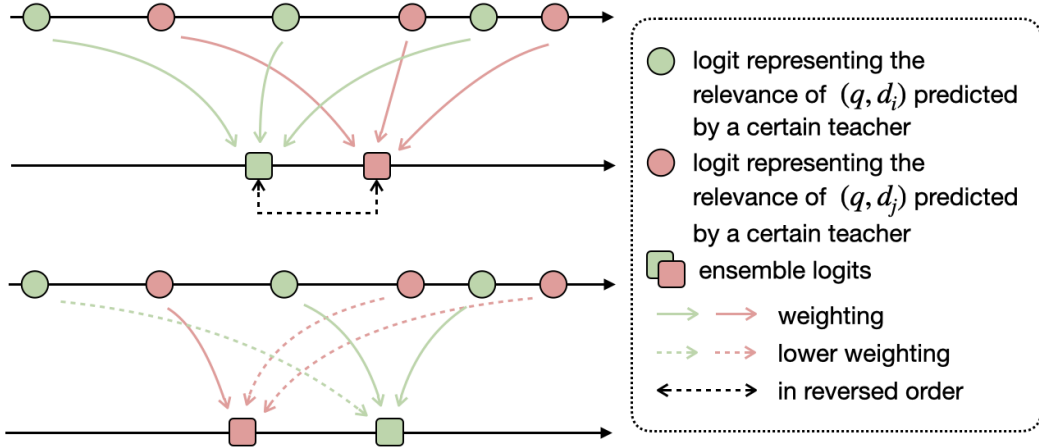


Figure 1: Procedures of PILE: 1) start with initializing the ensemble distillation logits with equal weights and 2) update a pair of resulting logits in reversed order by reassigning the weights of teachers.

reversed order or a negative pair and consider the ensemble logits have been biased. We modify the biased ensemble logits by reassigning to zero the weight of teachers responsible for the reversed order error. The reassignment rule can be formulated as follows:

$$w_k(q, d_i) = \begin{cases} 0, & f_k(q, d_i) < e^{(t)}(q, d_i) \\ 1, & \text{otherwise} \end{cases}$$

$$w_k(q, d_j) = \begin{cases} 0, & f_k(q, d_j) > e^{(t)}(q, d_j) \\ 1, & \text{otherwise} \end{cases}$$

where we assume the docs pair (d_i, d_j) is in reversed order and label $y_i > y_j$ for ease of explanation. Then, we update the ensemble logits with an update rate λ :

$$\tilde{e}^{(t+1)}(q, d) = \frac{1}{Z(q, d)} \sum_i w_k(q, d) f_k(q, d)$$

$$Z(q, d) = \sum_k w_k(q, d)$$

$$e^{(t+1)}(q, d) = (1 - \lambda)e^{(t)}(q, d) + \lambda\tilde{e}^{(t+1)}(q, d)$$

We repeat the updating in an iterative process until the magnitude of the ensemble logits of each pair of docs is consistent with their labels or it reaches the maximum iteration number. We use the final ensemble logits to perform knowledge distillation for the student model.

4 Experiments

To investigate the effectiveness of our proposed method, we conduct offline experiments with baseline models and deploy our proposed model in the

Data	#Query	#Query-Doc Pair
log data	635,420,390	2,970,692,361
train data	432,410	8,794,863
test data	12,044	289,835

Table 1: Dataset statistics

real-world production environment. In this section, we report the details of the experiment setups, datasets we used, evaluation metrics, the results of the experiments, and the case study.

4.1 Datasets

The datasets on which we pre-train, finetune, and evaluate our proposed method are collected from the Baidu search engine. For the pre-training stage, we collect a large-scale unlabeled dataset (log data) by means of the anonymous search log. The dataset contains 2,970,692,361 query-document pairs. As regards finetuning stage, queries and documents are collected from the search pipelines and manually labeled on Baidu’s crowdsourcing platform, where a group of hired annotators assigned an integer label range from 0 to 4 to each query-document pair, representing their relevance as {bad, fair, good, excellent, perfect}. We repeat the same process for the test set. The dataset information is summarized in Table 1.

4.2 Training details

We use a 12-layer Transformer (Vaswani et al., 2017) structure with 768 hidden sizes and 12 attention heads as the backbone of teacher models and a 6-layer Transformer structure with 768 hid-

den sizes and 12 attention heads as student models, where the parameters are randomly initialized, pre-trained and finetuned on the datasets described in section 4.1.

In order to obtain multiple teachers with similar performance and some differences for ensemble knowledge distillation, we use the same pre-trained checkpoint and finetune it on different samplings of the total train data. Specifically, the teacher T_1 is finetuned on the whole train data, and teacher T_2 and T_3 are finetuned on 80% of the whole train data. The intuition behind this is that we want all the teachers to perform relatively well but not behave as the same one otherwise the ensemble of three teachers may degenerate into a single model, which will compromise the benefits of the ensemble knowledge distillation. As a result, T_1 performs best on the test set and the other two teachers have a competitive performance as T_1 . The results of three teachers on the test set are shown in Table 2.

In the training procedure, we use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. For both 12-layer and 6-layer models, we set the learning rate as $2e-5$, the batch size as 64, and the warm-up step as 1000. In the PILE procedure, the maximum number of iterations is set to $|D_q|^{\frac{3}{2}}$ where $|D_q|$ is the size of documents set D_q related to a given query q and the update rate λ is set to 0.9. In the knowledge distillation stage, we set the warm-up step as 1000, the learning rate as $2e-5$ and the batch size as 1024.

4.3 Evaluation Metrics

The evaluation metrics we used in the experiments are as follows.

The **Positive-Negative Ratio (PNR)** measures the consistency between the golden labels and the scores output by models. For a given query q and a list of N associated documents ranked by model, the PNR can be calculated by this formulation:

$$PNR = \frac{\sum_{i,j \in [1,N]} I\{y_i > y_j\} I\{f(q, d_i) > f(q, d_j)\}}{\sum_{i,j \in [1,N]} I\{y_i > y_j\} I\{f(q, d_i) < f(q, d_j)\}},$$

where $I\{\cdot\}$ is the indicator function, taking the value 1 if the internal statement is true or 0 otherwise. For the test set that contains a good many queries, we average PNR values over all queries.

The **Discounted Cumulative Gain (DCG)** (Järvelin and Kekäläinen, 2000) is a widely used metric that evaluates the ranking result of search engines. More specifically, DCG is calculated as a

weighted sum of the document’s relevance degree G_i at each position i , where the weight is assigned according to the document’s position in the ranking results:

$$DCG = \sum_i \frac{G_i}{\log_2(i+1)}$$

The **Interleaving** (Chapelle et al., 2012; Chuklin et al., 2015) is extensively used for comparing a new system with the base system in industrial information systems evaluation. The results of two systems are interleaved and presented together to the end users, whose clicks would be credited to the system that provides the corresponding results. The gain of the new system A over the base system B can be denoted as Δ_{AB} :

$$\Delta_{AB} = 0.5 * \frac{wins(A) - wins(B)}{wins(A) + wins(B) + ties(A, B)},$$

where $wins(A)$ (or $wins(B)$) counts the number of times when the results produced by system A (or B) are more preferred than the other system for a given query and $ties(A, B)$ counts the number of times when the two systems are tied.

We also conduct a comparison called **Good vs. Same vs. Bad (GSB)** between two systems by inviting professional annotators to estimate which system produced a greater ranking result for each given query (Zhao et al., 2011). The gain of a new system can be formulated as:

$$\Delta_{GSB} = \frac{\#Good - \#Bad}{\#Good + \#Bad + \#Same},$$

where $\#Good$ (or $\#Bad$) denotes the number of queries that the new (or base) system provides better ranking results and $\#Same$ for the number of results that are equal in quality.

4.4 Offline Experimental Results

We conduct several comparison experiments to verify our proposed method. The models in the offline comparison experiments include:

- **Base:** We use an ERNIE-based ranking model as our base model, which is finetuned with a pairwise loss using human-labeled query-document pairs without any guidelines from teachers;
- **single-KD:** In this setting we add knowledge distillation loss when training the base model using the teacher that performs best on the test set;

Method	PNR	Improvement
Teacher1	3.21	-
Teacher2	3.20	-
Teacher3	3.19	-
Base	3.11	-
+ single-KD	3.15	+1.29%
+ AE-KD	3.16	+1.61%
+ PILE-KD	3.18	+2.25%

Table 2: Offline comparison of the proposed methods.

- **AE-KD**: Instead of using the single teacher, this variant uses an ensemble of 3 teachers with averaged weight to perform knowledge distillation;
- **PILE-KD**: When performing knowledge distillation, PILE-KD uses human-annotated labels with the help of the PILE algorithm to conduct a dynamic weighting process for the ensemble of 3 teachers.

The results of each model are shown in Table 2 with the improvement compared to the base model. We also report the performance of the teachers used in knowledge distillation. As we expected, all the distilled models consistently outperform finetuned base model thanks to teacher models’ guidance and regularization. And besides, using an ensemble of teachers gains further promotion than the single teacher distillation. After ensembling multiple teachers by averaging distillation logits, the PNR reaches 3.16, exceeding the base by 1.61%. This shows that the remission from biased distillation by the cooperation of multiple teachers improves students in semantic matching. Moreover, by applying the PILE algorithm, we can see that the student can beat the base model by a large margin w.r.t PNR, where the value is improved to 3.18 by 2.25% improvement. It shows the effectiveness of dynamic reduction of biased teachers’ weight in the ensemble process.

4.5 Online Experimental Results

To investigate the effectiveness of our proposed method in the real production environment, we deploy the proposed model in Baidu Search, a widely used Chinese search engine, and conduct online experiments for comparison.

The results are presented in Table 3, which comprises the performance comparison regarding ΔDCG , ΔGSB , and Δ_{AB} . We consider the

	Random	Tail
ΔDCG	+0.10%	+0.27%
ΔGSB	+3.70%	+1.62%

	Query Type		Query Length	
	Random	Tail	short	long
Δ_{AB}	+0.022%	+0.029%	+0.01%	+0.039%

Table 3: Online comparison of the proposed methods.

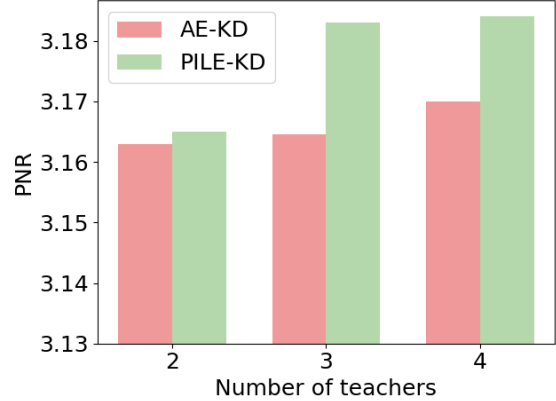


Figure 2: The effect of the number of teachers.

queries from the perspective of types and lengths in the search log. The tail queries and long queries are the queries whose search frequency is lower than 10 times per week or whose length is greater than 10 respectively. Since the heterogeneous search queries follow long-tail distributions, the tail queries make up a significant part of the queries in the search engine. As we can see the proposed method improves the performance of the online ranking system consistently. Particularly, we can observe that the gains of tail queries in the ΔDCG and ΔGSB for our method are 0.27% and 1.62% respectively. Compared with AE-KD, PILE-KD can enable students to retain the ability of teachers as much as possible, and the improvement on long-tail queries also confirms this.

4.6 Ablation Studies

To illustrate the detailed effects of the proposed algorithm, we take a deep insight into the contributions of each setting.

Number of teachers. We first focus on the effect of the number of teachers and the results are shown in Figure 2. As expected, the PNRs under both AE-KD and PILE-KD settings increase with the number of teachers and consistently outperform

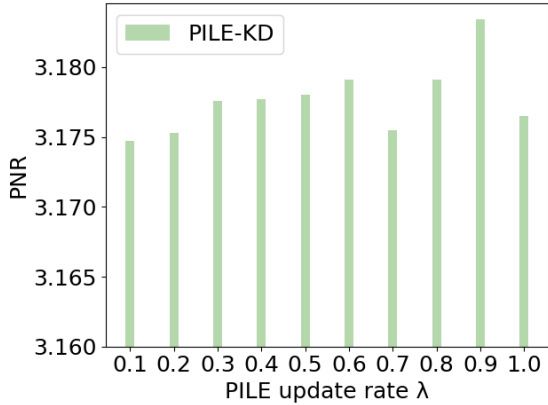


Figure 3: The effect of update rate λ in PILE.

the result of the single-KD model, showing the benefits of using multiple teachers for distillation. Besides, the performance under PILE-KD settings has always been better than it under AE-KD settings, showing the benefits of reducing noise and bias with the help of the label information annotated by human experts in the process of ensemble knowledge distillation.

PILE update rate. We further examine the effect of the PILE update rate λ . As shown in section 3.3, the update rate λ controls the smoothness of two successive iterations. As we can see in Figure 3, the PNR result increases with the λ ranging from 0.1 to 1.0, until reaching the apex at 0.9. The larger update rate λ makes the reliable teachers get more weight in a PILE iteration, resulting in more dependable soft targets. The results further prove that the distillation process can benefit from PILE iteration.

4.7 Case Study

To illustrate the effect of the PILE algorithm in the ensemble knowledge distillation concretely, we show a case that corrects the ensemble logits of two documents that are in reversed order.

As we can see in Table 4, two documents that are related to the same query are labeled by annotators as 0 and 3 respectively, representing their relevance with the query. The teachers' predictions of the relevance are listed in the table as $T_1 \sim T_3$. In the last two rows of the table, we show the ensemble results using averaged weight (AE) and the PILE method respectively.

To get the ensemble logits for knowledge distillation, AE takes the mean of teachers' predictions. However, influenced by individual teachers, the result ensemble logits of the two documents are

query	北京驾校教练一个月能挣多少钱? (How much does a Beijing driving school coach earn a month?)	
doc	北京私人教练一个月能挣多少钱? (How much does a Beijing personal trainer make a month?)	驾校教练工资一个月多少? 百度知道 (How much is the salary of a driving school coach per month? Baidu Knows)
label	0	3
T_1	0.0589	0.0271
T_2	0.1923	0.0331
T_3	0.1057	0.0983
AE	0.1190	0.0528
PILE	0.0590	0.0981

Table 4: The teachers' logits and ensemble logits for two documents.

contrary to their golden labels. In other words, the document with higher relevance is scored lower than the irrelevant one after the ensemble process, which will confuse the student in the knowledge transfer process. Benefiting from the PILE iteration, the teachers consistent with the golden label are assigned more weight. The resulting soft targets not only retain the knowledge that transfers from teachers but also are integrated with the label information annotated by human experts, which is more promising for knowledge distillation.

5 Conclusion

In this work, we propose an easy-to-implement approach to multi-teacher distillation for large-scale ranking models. Our algorithm ensembles multi-teacher logits supervised by human-annotated labels in an iterative way. We conduct the offline experiments as well as deploy our methods in an online commercial search system which demonstrates its superiority.

Acknowledgements

The authors would like to thank the colleagues at Baidu Inc. for their constructive suggestions that helped improve the paper, and hope everything goes well with their work. The authors are also indebted to the anonymous reviewers for their valuable comments and suggestions on the paper.

References

- Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.
- Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136.
- Olivier Chapelle, Thorsten Joachims, Filip Radlinski, and Yisong Yue. 2012. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30:6:1–6:41.
- Aleksandr Chuklin, Anne Schuth, Ke Zhou, and Maarten De Rijke. 2015. A comparative analysis of interleaving methods for aggregated search. *ACM Trans. Inf. Syst.*, 33(2).
- William S Cooper, Fredric C Gey, and Daniel P Dabney. 1992. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 198–210.
- Bucila Cristian, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- 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*.
- Shangchen Du, Shan You, Xiaojie Li, Jianlong Wu, Fei Wang, Chen Qian, and Changshui Zhang. 2020. Agree to disagree: Adaptive ensemble knowledge distillation in gradient space. *Advances in Neural Information Processing Systems*, 33:12345–12355.
- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015a. Distilling the knowledge in a neural network. *Computer Science*, 14(7):38–39.
- Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015b. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).
- Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.
- Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.
- Kalervo Järvelin and Jaana Kekäläinen. 2000. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- X. Jiao, Y. Yin, L. Shang, X. Jiang, and Q. Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*.
- Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- 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*.
- 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*.
- Hang Li. 2014. Learning to rank for information retrieval and natural language processing. *Synthesis lectures on human language technologies*, 7(3):1–121.
- Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. Dynamic knowledge distillation for pre-trained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Ping Li, Qiang Wu, and Christopher Burges. 2007. Mcrank: Learning to rank using multiple classification and gradient boosting. *Advances in neural information processing systems*, 20.
- Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. 2021. Pretrained transformers for text ranking: Bert and beyond. *Synthesis Lectures on Human Language Technologies*, 14(4):1–325.
- Yiding Liu, Weixue Lu, Suqi Cheng, Daiting Shi, Shuaiqiang Wang, Zhicong Cheng, and Dawei Yin.

2021. Pre-trained language model for web-scale retrieval in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3365–3375.
- Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198.
- Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.
- Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*.
- Morteza Mousa Pasandi, Mohsen Hajabdollahi, Nader Karimi, and Shadrokh Samavi. 2020. Modeling of pruning techniques for deep neural networks simplification. *arXiv preprint arXiv:2001.04062*.
- 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.
- 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*.
- S. Sun, Y. Cheng, Z. Gan, and J. Liu. 2019a. Patient knowledge distillation for bert model compression. 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)*.
- Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019b. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.
- Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, 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. *Advances in neural information processing systems*, 30.
- 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 *International Conference on Learning Representations*.
- Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pre-trained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.
- Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.
- Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. 2021. One teacher is enough? pre-trained language model distillation from multiple teachers. *arXiv preprint arXiv:2106.01023*.
- Canwen Xu, Wangchunshu Zhou, Tao Ge, Furu Wei, and Ming Zhou. 2020. Bert-of-theseus: Compressing bert by progressive module replacing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. 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 3490–3496.
- Shan You, Chang Xu, Chao Xu, and Dacheng Tao. 2017. Learning from multiple teacher networks. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Shiqi Zhao, Haifeng Wang, Chao Li, Ting Liu, and Yi Guan. 2011. Automatically generating questions from queries for community-based question answering. In *Proceedings of 5th international joint conference on natural language processing*.
- Zhaohui Zheng, Keke Chen, Gordon Sun, and Hongyuan Zha. 2007. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294.
- Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. 2021. Pre-trained language model based ranking in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4014–4022.

A Comprehensive Evaluation of Biomedical Entity-centric Search

Elena Tutubalina

Insilico Medicine Hong Kong
elena@insilicomedicine.com

Zulfat Miftakhtudinov

Insilico Medicine Hong Kong
zulfat@insilicomedicine.com

Vladimir Muravlev

Insilico Medicine Hong Kong
v.muravlev@insilicomedicine.com

Anastasia Shneyderman

Insilico Medicine Hong Kong

a.shneyderman@insilicomedicine.com

Abstract

Biomedical information retrieval has often been studied as a task of detecting whether a system correctly detects entity spans and links these entities to concepts from a given terminology. Most academic research has focused on evaluation of named entity recognition (NER) and entity linking (EL) models which are key components to recognizing diseases and genes in PubMed abstracts. In this work, we perform a fine-grained evaluation intended to understand the efficiency of state-of-the-art BERT-based information extraction (IE) architecture as a biomedical search engine. We present a novel manually annotated dataset of abstracts for disease and gene search. The dataset contains 23K query-abstract pairs, where 152 queries are selected from logs of our target discovery platform and PubMed abstracts annotated with relevance judgments. Specifically, the query list also includes a subset of concepts with at least one ambiguous concept name. As a baseline, we use off-the-shelf Elasticsearch with BM25. Our experiments on NER, EL, and retrieval in a zero-shot setup show the neural IE architecture shows superior performance for both disease and gene concept queries.

1 Introduction

The amount of text data being produced is overwhelming, especially in biomedicine; PubMed¹ covers over 33 million articles from biomedical and life sciences journals and other texts, with about 1.5 million added each year. Meanwhile, many of these articles are about specific entities (e.g. proteins, diseases, chemicals), i.e., entity-centric. In general, entities are central to many search queries; e.g., (Guo et al., 2009) demonstrated that 71% of search queries contained named entities, while (Xiong et al., 2017) found that more than half of the traffic in the Allen Institute’s scholar search engine is about research concepts.

¹<https://pubmed.ncbi.nlm.nih.gov>

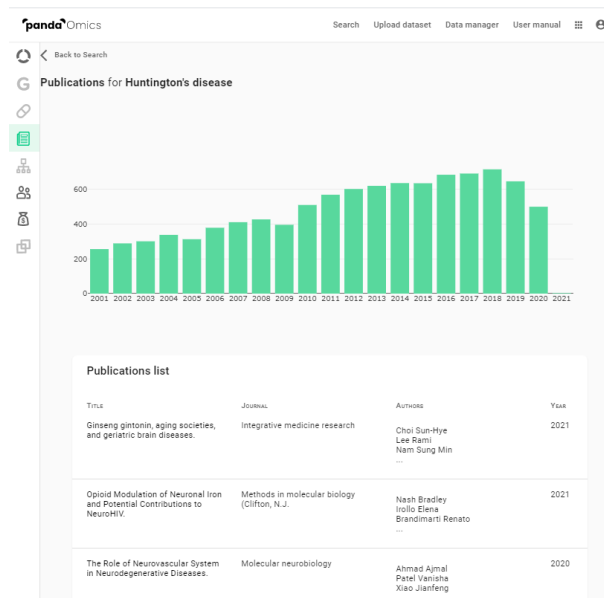


Figure 1: Publication page for the ‘Huntington disease’ query in our target discovery platform PandaOmics (<https://pandaomics.com/>).

The use of automatic natural language processing (NLP) methods is imperative for information retrieval (IR) or information extraction (IE) from a large volume of biomedical texts. Several efforts have been made in the past years on entity extraction from scientific publications (Kim et al., 2013; Lee et al., 2016; Allot et al., 2018; Mohan et al., 2018, 2021; Wang and Lo, 2021). For example, Biomedical Entity Search Tool (BEST) uses a dictionary-based indexing strategy to extract ten types of biomedical entities including genes, diseases, drugs, and chemical compounds (Lee et al., 2016), while (Kim et al., 2013; Mohan et al., 2021) adopt machine learning for disease and gene extraction and linking. However, recent works on Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) showed that the generalization ability of BERT-based named entity recognition (NER) and entity linking (EL) models is influenced by domain shift or whether the test en-

tity/term has been seen in the training set (Miftahudinov et al., 2020; Tutubalina et al., 2020; Kim and Kang, 2022). Recently, (Soni and Roberts, 2021) compared two commercial search engines with academic prototypes evaluated in the TREC-COVID challenge (Roberts et al., 2020; Voorhees et al., 2021). Their evaluation showed that commercial search engines from Amazon (CORD-19 Search) and Google (COVID-19 Research Explorer) fail to outperform decades-old IR approaches. In particular, the best run (from sabir) was achieved by a SMART system (Buckley, 1985) and used no machine learning or biomedical knowledge. A similar observation has been made for general-domain information retrieval (Thakur et al., 2021), where more efficient approaches e.g. based on dense or sparse embeddings can substantially underperform traditional lexical models like BM25 (Robertson and Zaragoza, 2009).

In this paper, we describe the design and evaluation of a BERT-based IE system as an entity-centric search engine for a target discovery platform PandaOmics². In particular, we seek to answer the following research question: considering near excellent performance on NER and EL (Miftahudinov et al., 2021; Lee et al., 2019), are there models capable of finding relevant publications for disease and gene queries from diverse biomedical subdomains as real-world applications? To help answer this question, we develop a novel search collection of PubMed abstracts for disease and gene queries with corresponding relevance judgments. We evaluate the IE pipeline with two trained BERT-based models for NER and EL and standard document retrieval model BM25 with off-the-shelf Elasticsearch software. We perform error analysis on the models' predictions to shed light on future work directions.

2 Dataset

This section describes our dataset, including queries, and the process of collecting relevance assessments. Table 1 shows statistics of our dataset.

2.1 Queries

In our target discovery platform PandaOmics, a user can enter a gene name or gene symbol like 'PSEN1' (ENSG00000080815) and retrieve all relevant publications and the associated diseases in-

²<https://pandaomics.com/>

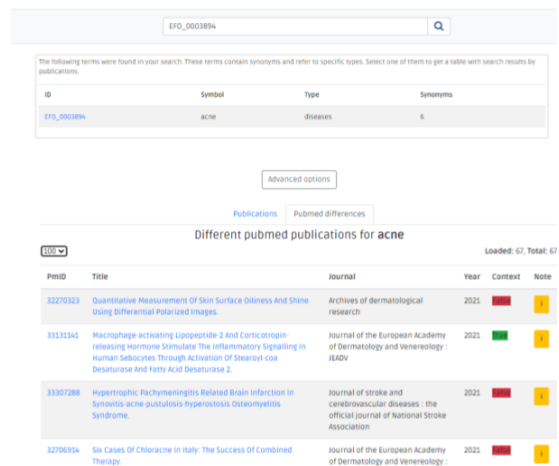


Figure 2: Task design in our in-house annotation tool with search by disease concept identifier. An annotator selects an abstract and choose one of three labels (relevant/true (green), nonrelevant/false (red), or doubtful (yellow)).

cluding Alzheimer's disease (EFO:0000249). An autocomplete feature displays suggestions from disease or gene dictionaries as user search terms. Conversely, the user can enter the disease name 'Alzheimer's disease' to retrieve publications for this concept and the associated targets. These associations are relying on Omics datasets and on a collection of AI-based scores that are based on molecular data and previously published text-based data (see (Ozerov et al., 2016) for more details). As a disease terminology source, we use an internal knowledge base that contains 15,051 concept unique identifiers (CUIs) based on an experimental factor ontology (EFO)³ (Malone et al., 2010). As a gene terminology source, we use an internal knowledge base with 28,227 CUIs from Ensembl (Hubbard et al., 2002). We recall that each concept consists of atoms (concept names); all of the atoms within a concept are synonymous (NLM, 2016). As test queries for our dataset, we use the most frequent queries from the platform's logs. These queries are disease CUIs and gene CUIs. In addition, our annotators selected a list of concepts with at least one ambiguous concept name (see Table 2 for examples).

2.2 Relevance Assessments

2.2.1 Pooling

Following standard practice of IR collection building, we employ a *pooling* approach (Lipani et al.,

³<https://www.ebi.ac.uk/efo/>

Subset	#queries	avg. number of texts per query		
		relevant label	nonrelevant label	doubtful label
Disease CUI	73	94.86	63.57	9.78
Gene CUI	79	109.39	21.62	5.93
Ambiguous	27	45.94	11.58	0.53
Total	152	102.41	41.76	7.78

Table 1: Summary of statistics of the proposed dataset.

CUI	Ambiguous concept name	Term	Reason	Comment
EFO_0000341	coad	chronic obstructive pulmonary disease	same_synonyms	abbreviation refers to another disease: colon adenocarcinoma (COAD)
EFO_1001998	crps	complex regional pain syndrome	same_synonyms	abbreviation refers to another disease: ‘Colorectal polyps’
EFO_1001998	crps	complex regional pain syndrome	same_synonyms	abbreviation refers to another disease: ‘chronic regional pain syndrome’
EFO_0000341	dops	chronic obstructive pulmonary disease	refers_to_another	abbreviation refers to another term: direct observation of procedural skills (DOPS)
EFO_0002508	parkinson’s disease	parkinson’s disease	refers_to_another	author’s surname
ENSG00000170345	fos	fos	refers_to_another	refers to fosfomycin

Table 2: A sample of concepts with at least one ambiguous concept name.

2016; Lipani, 2016; Hasibi et al., 2017; Thakur et al., 2021), and combine retrieval results from two main sources:

1. we obtained retrieval results from Elasticsearch; see Sect. 3.2 for the description of this system. Results are pooled from these runs up to depth 100.
2. we obtained retrieval results from PubMed. Results are pooled from these runs up to depth 100, excluding abstracts from the first system.

The final assessment pool contains 23,099 query-abstract pairs (152 abstracts per query on average).

2.2.2 Collecting Relevance Judgments

For each query-abstract pair, we collected the relevance judgments by 2 annotators with biomedical degrees using an in-house annotation tool (Fig. 2). An expert annotator with Ph.D. in biology created a list of queries from logs of our target discovery platform PandaOmics. All annotators are paid biologists in the company. An expert annotator wrote annotation guidelines and educated annotators.

Each annotator selected a disease or gene query from the list of selected identifiers, an abstract with

information about the publication year and journal. Abstracts were presented in random order. Annotators were then asked to: (i) judge relevance on a 3-point scale: “relevant”, “nonrelevant”, or “doubtful”, and (ii) categorize the reason for relevance/nonrelevance.

We note that annotators were asked to consider EFO hierarchy during relevance annotation for disease queries. According to the annotation guidelines, only the synonyms belonging to the required level of the hierarchy are relevant. Those terms that are higher in the hierarchy are “wider terms”, and those that are lower represent a “narrower case”. E.g., while annotating a text for the “prostate adenocarcinoma” query, “prostate cancers” is wider than the term of interest; for the “prostate cancer” query, the “prostate adenocarcinoma” is narrower than the term of interest. Further, we provide a summary of guidelines illustrated with examples.

Relevance The publication relevance to a gene/disease can be determined as true when the gene/disease of interest (its main name or any synonym) is present in the same meaning in an abstract. The term in the abstracts should belong to a disease/gene ontology (and not to any other category,

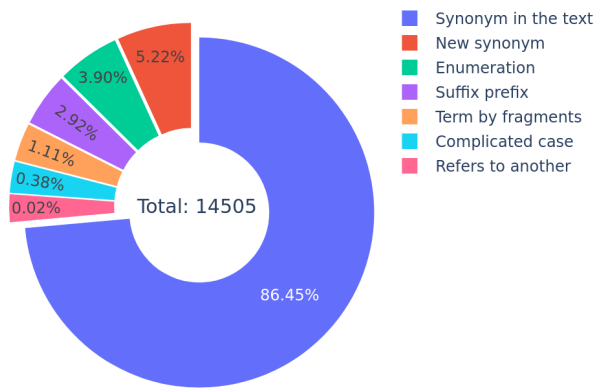


Figure 3: Statistics of Relevance reasons.

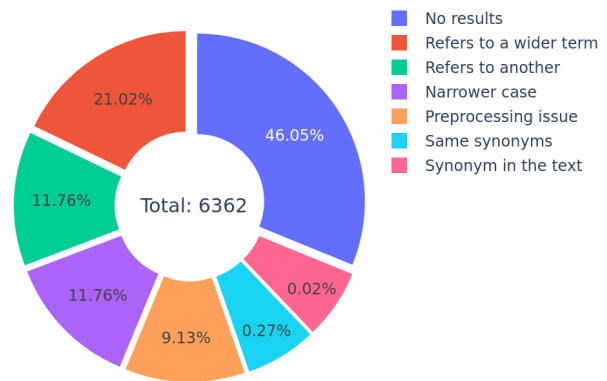


Figure 4: Statistics of Nonrelevance reasons.

e.g. name of a clinical trial, institution, foundation, etc). In particular, there are six reasons for relevance:

1. **synonym in text** – one of the synonyms is precisely present in an abstract;
2. **new synonym** – new synonym for the term of interest, which is absent in our synonyms list, was found;
3. **term by fragments** - an entity is annotated by several fragments of text if: (i) a term is either from the disease of gene ontology; (ii) both fragments are in the same sentence; (iii) the parts of the term are logically connected (according to the author's logic). E.g., the text "...secondary *diabetic* complications, such as *retinopathy*, neuropathy, and nephropathy" (pmid 33109031) should be annotated as TRUE for "diabetic retinopathy";
4. **enumeration** - an entity is annotated by fragments which are separated only with punctuation marks or conjunctions. E.g., the text "asthma-wheezing" (pmid 33276583) should be annotated as a true for both "asthma" and "wheezing", while "AKT1-mTORC1 Axis" (pmid 32404972) should be annotated as TRUE for "AKT1";
5. **suffix/prefix** - an entity was annotated as a part of a word with a suffix/prefix. E.g., we annotate "obesity-induced NAFLD" as a match for "obesity" and add "-induced" as a suffix (we note that there is no "obesity-induced NAFLD" term in the ontology);
6. **complicated case** – a term is encountered in abstract by fragments separated in different

sentences, and there is a logical link between them.

Detailed distribution of relevance reasons are given in Fig. 3.

Nonrelevance Nonrelevance of a gene/disease is determined as either no link between the gene/disease and a publication abstract or a wrongly identified relation. The first means the gene/disease is not mentioned in an abstract. The second means that gene/disease is incorrectly linked to an abstract because of one of the following six reasons:

1. **no results** – no results for the term of interest were found in a publication;
2. **refers to another** – gene/disease name (or its abbreviation) is a synonym of some other term, or has some other meanings, which are outside of the ontology (e.g., abbreviation COAD for colon adenocarcinoma refers to another term "anaerobic co-digestion (co-AD)", an abbreviation for Non-alcoholic steatohepatitis refers to another term "Nash equilibria");
3. gene/disease name (or its abbreviation) refers to another term within the ontology (gives *collisions*) because of: (i) **same synonyms** (e.g., abbreviation COAD for Chronic obstructive pulmonary disease refers to another disease "colon adenocarcinoma"); (ii) **refers to a wider term** – publication abstract was found by a wider disease term, which refers not only to a disease of interest, and may give additional non-relevant results (e.g, colon cancer is wider term for colon adenocarcinoma); (iii) **narrower case** – publication abstract was

found by more specific term (e.g., Alzheimer’s disease is a narrower case for neurodegenerative disease); (iv) **preprocessing issue** - either ignored punctuation mark (“*background: retinopathy*”, “*ER-breast cancer*”) or is a part of a longer term (“*Non-small cell lung carcinoma*”, “*Traf2- and Nck-interacting kinase*”).

Detailed distribution of nonrelevance reasons is given in Fig. 4.

We note that our definition of nonrelevance differs from Pubmed search primarily because of the consideration of the concept hierarchy. PubMed search uses Best Match (Fiorini et al., 2018) trained on the user-click information from PubMed search logs. We believe that distinguishing more narrow concepts from broader ones is crucial for target discovery objectives.

Doubtful This category includes publications that mention disease/gene of interest only in keywords/MeSH terms **without an abstract match**. PubMed articles are manually associated with author keywords and MeSH (Medical Subject Headings) (Lipscomb, 2000) as standardized keywords. The reasons for this label are the same as for the relevance label with **synonym in MeSH/keywords** and excluding the “complicated case” category. In 97.65% and 1.6% cases, the annotator associated texts with the synonym in MeSH/keywords and new synonym reasons, respectively.

In 91% and 80% of pairs, two annotators agreed on a relevance label and decision reasons, respectively. When annotators disagreed, the expert annotator was asked to decide whether the relevance labels among with reasons selected by one of the annotators were in fact correct. After this procedure, we obtained the dataset for entity search with 73 disease queries, 79 gene queries, and 23,099 annotated query-entity pairs.

3 Models

The goal of our work is to evaluate retrieval models in a zero-shot setup, with no available training data to train the IR system.

3.1 BERT-based IE pipeline

In our work, we have focused on the extraction of two entity types: disease and gene. Though, we design our IE system with the simplicity of scaling to new entities in mind. The system consists of pipelines, each for a different entity type.

The pipelines incorporate two sub-modules: (i) NER sub-module; (ii) EL sub-module. These sub-modules are applied successively. The first one extracts entities of interest the second one links extracted entities with concepts from given knowledge bases. Taken all together it means that the processing of different types of entities is independent and could be trained and applied separately. As a pretrained transformer model, we use BioBERT base v1.1. (Lee et al., 2019).

Named Entity Recognition In this paper, for reproducibility reasons, we decided to analyze models trained on publicly available academic datasets. Specifically, we train BioBERT on combination of NCBI and CDR Diseases datasets (Doğan et al., 2014; Li et al., 2016) for disease entities and on DrugProt dataset (Miranda et al., 2021) for gene entities. To join the NCBI and CDR Disease datasets, we utilized predefined train/test subsets and combined the datasets within these splits. Thus, the train part of NCBI was combined with the CDR Disease train sets. A similar procedure was carried out to obtain the test part of the combined dataset. We adopted model training hyper-parameters from (Lee et al., 2019). Our model achieves 88.43% and 90.39% of the F-measure on official test sets of disease and gene entities, respectively.

Entity Linking For linking extracted entities to corresponding concepts from dictionaries, we employ state-of-the-art Drug and disease Interpretation Learning with Biomedical Entity Representation Transformer (DILBERT) (Miftahutdinov et al., 2021). This model is based on metric learning and negative sampling, specifically, triplet constraints. Given an entity mention m , a positive concept name c_g and a negative concept name c_n , triplet loss tunes the network such that the distance between m and c_g is smaller than the distance between m and c_n . Details on overall architecture, configuration, hyperparameter search, and evaluation strategies are presented in (Miftahutdinov et al., 2021). The code is publicly available at <https://github.com/insilicomedicine/DILBERT>. We note that the advantage of DILBERT architecture is the ability to search for the closest concept in a different terminology without retraining the model (cross-terminology use).

Similar to NER, we train models on publicly available academic datasets: CDR Diseases (Li

et al., 2016) and BC2GN Genes (Morgan et al., 2008). The models are evaluated on *refined* test sets without entity overlap between train/test sets from (Tutubalina et al., 2020). These sets are publicly available at <https://github.com/insilicomedicine/Fair-Evaluation-BERT>. Our model achieves 75.8% and 82.4% of accuracy on the refined test sets of diseases and genes, respectively.

Details on models’ configurations, speed performance and system deployment are presented in Appendices A and B.

3.2 Elasticsearch BM25

We utilized a popular search engine framework Amazon Elasticsearch/OpenSearch Service⁴ that uses OpenSearch v.1.0⁵. OpenSearch is a fork of open source Elasticsearch 7.10⁶. OpenSearch uses BM25 (Robertson and Zaragoza, 2009) to calculate relevance scores. BM25 is a commonly-used bag-of-words retrieval function based on token-matching between two high-dimensional sparse vectors with TF-IDF token weights. We note that (Thakur et al., 2021) recently showed that many approaches with sparse, dense late-interaction architectures outperform BM25 on in-domain evaluation, yet perform poorly on zero-shot setup.

4 Evaluation

For evaluation, we use precision, recall, and F-measure. We calculate the precision as a fraction of relevant documents among all retrieved documents. As well the recall is calculated as a fraction of relevant documents from all possibly relevant documents in the dataset. For experiments, we use query-document pairs with relevant and nonrelevant labels excluding the doubtful category.

Tables 3 and 4 present the performance of the BERT-based pipeline compared to BM25 on the full set of queries and the subset of concept with ambiguous names, respectively. Several observations can be made based on Tables 3 and 4. First, the BERT-based system outperformed BM25 on both sets of the dataset and both types of entities. As expected, the performance difference between the two models is larger on the subset with ambiguous concept names. Third, for the BERT-based pipeline, precision is higher than recall.

⁴<https://aws.amazon.com/opensearch-service/>

⁵<https://opensearch.org/>

⁶<https://www.elastic.co/>

Model	P	R	F
Queries with Disease CUIs			
BERT-based	93.97	84.41	88.93
Elasticsearch BM25	82.19	83.33	82.76
Genes			
BERT-based	92.24	85.45	88.71
Elasticsearch BM25	89.92	79.93	84.63
Both			
BERT-based	92.99	84.99	88.81
Elasticsearch BM25	86.23	81.44	83.77

Table 3: IR metrics on the full set of queries.

Model	P	R	F
Queries with Disease CUIs			
BERT-based	97.72	93.81	95.73
Elasticsearch BM25	75.67	96.72	84.91
Genes			
BERT-based	93.02	93.85	93.43
Elasticsearch BM25	79.58	68.88	73.85
Both			
BERT-based	94.9	93.83	94.37
Elasticsearch BM25	77.59	80.39	78.96

Table 4: IR metrics on the subset of queries with ambiguous concepts.

In addition, we investigate search precision further by developing a dataset for out-of-domain abstract detection. Approximately 30,000 records are included in the PubMed journal list. These journals publish papers not only about biological entities, but also on cultural topics, economics and econometrics, artificial intelligence, law, linguistics and language, and so on (out-of-domain categories for us). Our expert annotator manually selected out-of-domain journals on which we expect the IE system to return *zero results*. We randomly select 58,790 abstracts from these journals, where each abstract includes at least one gene of disease concept retrieved by Elasticsearch. In 90% of these abstracts, the BERT-based system did not find any entities.

Error Analysis For error analysis of the BERT-based IE system, we reviewed a sample of 152 false positive (FP) documents and 168 false negative (FN) results. Table 5 provides summary on error categories for FPs. As shown in Table 5, the most frequent category of errors (58%) is related to the ontology hierarchy. Wider cases can also be attributed to a gene when the gene family is mentioned (e.g., Akt (there are Akt1/2/3), ERK

Reason	N	%
wider term	88	58
refers to another	34	22
synonym in MeSH/keywords	17	11
same synonym	11	7
preprocessing issue	1	1
synonym in the text	1	1

Table 5: Error analysis of IR results on the false positive sample (152 texts).

Reason	Model	N
not found	-	48
largest text span exists	NER	23
not recognized	NER	6
abbreviation	NER	5
wrong recognition	NER	1
wrong mapping	EL	2
largest text span rule/wrong mapping	NER/EL	15

Table 6: Error analysis of NER and EL predictions on the false negative (FN) sample (100 texts).

(there are ERK1/2)). For FNs, 60% of errors (100 abstracts) fell into the synonym in the text category. These documents were additionally analyzed to detect which model (NER or EL) predicted incorrectly (see Table 6). As shown in Table 6, in 23% cases, the NER model predicts a shorter entity which is also known as a boundary problem. E.g., in the text “external validation of the Nonalcoholic [Steatohepatitis]_{predicted} Scoring System in patients” (pmid 33248101) Nonalcoholic Steatohepatitis was mapped to just Steatohepatitis due to NER predictions. Mapping errors are often related to the presence of numbers in gene names or abbreviations. E.g., in a text “orphan nuclear receptor [Nr4a1] mediates perinatal neuroinflammation” (pmid 32606386) entity *Nr4a1* mapped to the *Nr4a2* gene instead. For FPs, we additionally analyze 22% of errors (34 abstracts) from the refers to another category. The NER and EL models cause errors in 16 and 11 documents, respectively.

5 Conclusion and Future Work

In this work, we present a comprehensive evaluation of a biomedical entity-centric search engine based on BERT models for disease and gene extraction and linking. This engine is a part of a target discovery platform, where users can return a list of relevant publications given a disease or

gene concept query. We evaluate BERT models on two information extraction tasks, entity-centric information retrieval, and out-of-domain abstract detection. Moreover, we present an error analysis for both retrieval and extraction tasks.

This work suggests several interesting directions for future research. We plan to conduct similar studies on other text sources such as full publication texts and patents. Moreover, we plan to expand the list of entity types with pathways and biological processes. To extract explicit associations between drug targets and diseases, we plan to add relation extraction/event detection models and study knowledge graph completion with novel disease-gene edges.

6 Ethics Statement

We outline potential ethical issues with our work below. First, our work focuses on a comprehensive evaluation of the information extraction pipeline for retrieval of relevant scientific texts given queries of disease and gene concepts. Consequently, the developed BERT-based models could reflect many domain-specific biases exhibited by language models. For example, (Sung et al., 2021) showed that predictions on factual triples tend to be highly biased towards a few objects (e.g., “headache”, “pain”, or “ESR1”). Since pretrained language models are used for initialization, it is possible to reflect biased patterns in open-world applications. Second, our NLP engine is a part of the target discovery platform PandaOmics which intend to identify targets (genes/proteins) through deep feature selection, causality inference, and de novo pathway reconstruction (Ozerov et al., 2016). We use the NLP engine to assess the targets’ novelty and disease association via the analysis of research publications. The imperfect completeness of the extracted information can be especially reflected in the small number of publications in the search results about rare diseases, making it difficult for subsequent analysis. Third, we use EFO and Ensembl as primary resources with disease hierarchy and concepts’ synonyms. For example, (Miftahudinov et al., 2021) demonstrated that degradation in the accuracy from the full disease dictionary to a 30% of the dictionary is significant for disease linking in clinical trials. Moreover, consistent description of these entities has numerous differing standards and opportune incorporation of new human disease terms and targets is still necessary.

References

- Alexis Allot, Yifan Peng, Chih-Hsuan Wei, Kyubum Lee, Lon Phan, and Zhiyong Lu. 2018. Litvar: a semantic search engine for linking genomic variant data in pubmed and pmc. *Nucleic acids research*, 46(W1):W530–W536.
- Chris Buckley. 1985. Implementation of the smart information retrieval system. Technical report, Cornell University.
- 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), Minneapolis, USA*, pages 4171–4186.
- Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.
- Nicolas Fiorini, Kathi Canese, Grisha Starchenko, Evgeny Kireev, Won Kim, Vadim Miller, Maxim Osipov, Michael Kholodov, Rafis Ismagilov, Sunil Mohan, et al. 2018. Best match: new relevance search for pubmed. *PLoS biology*, 16(8):e2005343.
- Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. 2009. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274.
- Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. Dbpedia-entity v2: a test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1265–1268.
- Tim Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen, L Clark, Tony Cox, J Cuff, Val Curwen, Thomas Down, et al. 2002. The ensembl genome database project. *Nucleic acids research*, 30(1):38–41.
- Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.
- Hyunjae Kim and Jaewoo Kang. 2022. How do your biomedical named entity recognition models generalize to novel entities? *Ieee Access*, 10:31513–31523.
- Jeongkyun Kim, Seongeun So, Hee-Jin Lee, Jong C. Park, Jung-jae Kim, and Hyunju Lee. 2013. [DigSee: disease gene search engine with evidence sentences \(version cancer\)](#). *Nucleic Acids Research*, 41(W1):W510–W517.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.
- Sunwon Lee, Donghyeon Kim, Kyubum Lee, Jaehoon Choi, Seongsoon Kim, Minji Jeon, Sangrak Lim, Donghee Choi, Sunkyu Kim, Aik-Choon Tan, et al. 2016. Best: next-generation biomedical entity search tool for knowledge discovery from biomedical literature. *PLoS one*, 11(10):e0164680.
- Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wieggers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.
- Aldo Lipani. 2016. [Fairness in information retrieval](#). In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16*, page 1171, New York, NY, USA. Association for Computing Machinery.
- Aldo Lipani, Mihai Lupu, and Allan Hanbury. 2016. The curious incidence of bias corrections in the pool. In *European Conference on Information Retrieval*, pages 267–279. Springer.
- Carolyn E Lipscomb. 2000. Medical subject headings (mesh). *Bulletin of the Medical Library Association*, 88(3):265.
- Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.
- James Malone, Ele Holloway, Tomasz Adamusiak, Misha Kapushesky, Jie Zheng, Nikolay Kolesnikov, Anna Zhukova, Alvis Brazma, and Helen Parkinson. 2010. Modeling sample variables with an experimental factor ontology. *Bioinformatics*, 26(8):1112–1118.
- Zulfat Miftahutdinov, Ilseyar Alimova, and Elena Tutubalina. 2020. On biomedical named entity recognition: experiments in interlingual transfer for clinical and social media texts. In *European Conference on Information Retrieval*, pages 281–288. Springer.
- Zulfat Miftahutdinov, Artur Kadurin, Roman Kudrin, and Elena Tutubalina. 2021. Medical concept normalization in clinical trials with drug and disease representation learning. *Bioinformatics*, 37(21):3856–3864.
- Antonio Miranda, Farrokh Mehryary, Jouni Luoma, Sampo Pyysalo, Alfonso Valencia, and Martin Krallinger. 2021. Overview of drugprot biocreative vii track: quality evaluation and large scale text mining of drug-gene/protein relations. In *Proceedings of the seventh BioCreative challenge evaluation workshop*.

- Sunil Mohan, Rico Angell, Nicholas Monath, and Andrew McCallum. 2021. Low resource recognition and linking of biomedical concepts from a large ontology. In *Proceedings of the 12th ACM conference on bioinformatics, computational biology, and health informatics*, pages 1–10.
- Sunil Mohan, Nicolas Fiorini, Sun Kim, and Zhiyong Lu. 2018. A fast deep learning model for textual relevance in biomedical information retrieval. In *Proceedings of the 2018 World Wide Web Conference*, pages 77–86.
- Alexander A Morgan, Zhiyong Lu, Xinglong Wang, Aaron M Cohen, Juliane Fluck, Patrick Ruch, Anna Divoli, Katrin Fundel, Robert Leaman, Jörg Hakenberg, et al. 2008. Overview of biocreative ii gene normalization. *Genome biology*, 9(S2):S3.
- NLM. 2016. [Umls glossary](#).
- Ivan V Ozerov, Ksenia V Lezhnina, Evgeny Izumchenko, Artem V Artemov, Sergey Medintsev, Quentin Vanhaelen, Alexander Aliper, Jan Vijg, Andreyan N Osipov, Ivan Labat, et al. 2016. In silico pathway activation network decomposition analysis (ipanda) as a method for biomarker development. *Nature communications*, 7(1):1–11.
- Kirk Roberts, Tasmee Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R Hersh. 2020. Trec-covid: rationale and structure of an information retrieval shared task for covid-19. *Journal of the American Medical Informatics Association*, 27(9):1431–1436.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: Bm25 and beyond](#). *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Sarvesh Soni and Kirk Roberts. 2021. An evaluation of two commercial deep learning-based information retrieval systems for covid-19 literature. *Journal of the American Medical Informatics Association*, 28(1):132–137.
- Mujeen Sung, Jinhyuk Lee, Sean Yi, Minji Jeon, Sungdong Kim, and Jaewoo Kang. 2021. [Can language models be biomedical knowledge bases?](#) In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4723–4734, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
- Elena Tutubalina, Artur Kadurin, and Zulfat Miftahutdinov. 2020. Fair evaluation in concept normalization: a large-scale comparative analysis for bert-based models. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6710–6716.
- Ellen Voorhees, Tasmee Alam, Steven Bedrick, Dina Demner-Fushman, William R Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2021. Trec-covid: constructing a pandemic information retrieval test collection. In *ACM SIGIR Forum*, volume 54, pages 1–12. ACM New York, NY, USA.
- Lucy Lu Wang and Kyle Lo. 2021. Text mining approaches for dealing with the rapidly expanding literature on covid-19. *Briefings in Bioinformatics*, 22(2):781–799.
- Chenyang Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pages 1271–1279.

A System deployment

Our system is packaged in a docker container, which is run by schedule. Since the container is self-contained any scheduler could be used. The pipeline of documents processing in service is as follows: (i) load to the local storage previously unlabeled documents from a database (ii) extract and link entities from documents using the BERT-based pipeline (iii) upload the labeled documents to the database. We store our documents in MongoDB (<https://www.mongodb.com>). The service is implemented substantially on python, with endpoints written in shell. To load/upload documents from/to MongoDB we use PyMongo library (<https://pymongo.readthedocs.io>). After the labeled documents are loaded to the MongoDB we utilize Elasticsearch as a search index. Customers of the drug discovery platform send concept CUI as a query, afterward, the backend retrieve all documents containing specified CUI and transfer them to the frontend.

B Configuration details and speed performance

For NER and EL encoders, we apply the fine-tuned on downstream task BioBERT v1.1 with 12 heads, 12 layers, 768 hidden units per layer, and a total of 110M parameters. We train our NER model using AdamW (Loshchilov and Hutter, 2018) optimizer for 20 epochs with a batch size equal to 48 and learning rate equal to 5e-5. The EL model is

trained with the same optimizer and learning rate for 5 epochs and batch size equal to 32. At the inference and training time, we restrict the length of the sequence up to 128 sub-tokens for entity recognition and up to 28 sub-tokens for linking.

For NER sub-module we use Huggingface python library (<https://huggingface.co>), for EL we apply sentence-transformers library (<https://www.sbert.net>). At the inference time, the EL model uses the FAISS library (Johnson et al., 2019) with GPU support for a fast nearest neighbor search by comparing vectors with Euclidean distance. Embeddings of all terminologies' concepts are indexed.

We note that deployed models are trained on in-house datasets with similar parameters and evaluation metrics that are not publicly available due to company policy.

We profiled retrieval speed on a server with Intel Xeon CPU E5-2660 2.00GHz and 256GB memory. First, we precomputed all embeddings for all concepts (500 thousand). On a single Nvidia TITAN X GPU, it takes about 7 minutes to compute all embeddings. Given that all embeddings are indexed on Nvidia TITAN X GPU using IndexFlatL2 index type 5 thousand documents processing takes 390 seconds, which is 0.08 seconds per document. Most of this time, specifically 359 seconds, is taken by the NER sub-module.

Domain Adaptation of Machine Translation with Crowdworkers

Makoto Morishita¹, Jun Suzuki², Masaaki Nagata¹

NTT Communication Science Laboratories, NTT Corporation¹

Tohoku University²

{makoto.morishita.gr, masaaki.nagata.et}@hco.ntt.co.jp

jun.suzuki@tohoku.ac.jp

Abstract

Although a machine translation model trained with a large in-domain parallel corpus achieves remarkable results, it still works poorly when no in-domain data are available. This situation restricts the applicability of machine translation when the target domain’s data are limited. However, there is great demand for high-quality domain-specific machine translation models for many domains. We propose a framework that efficiently and effectively collects parallel sentences in a target domain from the web with the help of crowdworkers. With the collected parallel data, we can quickly adapt a machine translation model to the target domain. Our experiments show that the proposed method can collect target-domain parallel data over a few days at a reasonable cost. We tested it with five domains, and the domain-adapted model improved the BLEU scores to +19.7 by an average of +7.8 points compared to a general-purpose translation model.

1 Introduction

Although recent Neural Machine Translation (NMT) methods have achieved remarkable performance, their translation quality drastically drops when the input domain is not covered by training data (Müller et al., 2020). One typical approach for translating such inputs is adapting the machine translation model to a domain with a small portion of in-domain parallel sentences (Chu and Wang, 2018). Such sentences are normally extracted from a large existing parallel corpus (Wang et al., 2017; van der Wees et al., 2017) or created synthetically from a monolingual corpus (Chinea-Ríos et al., 2017). However, the existing parallel/monolingual data may not include enough sentences relevant to the target domain.

There is a real-world need for a method that can adapt a machine translation model to any domain. For example, users reading or writing in such specific fields as scientific, medical or patent domains,

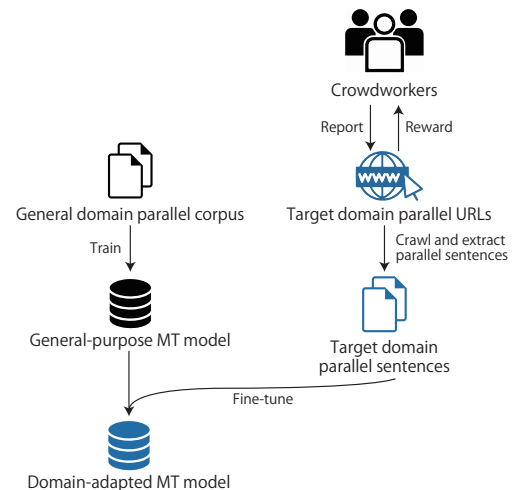


Figure 1: Overview of proposed domain-adaptation method with crowdworkers who collected URLs that included parallel sentences of target domain. We then fine-tuned a general-purpose model with the collected target domain parallel sentences. See Section 3 for details.

may experience satisfaction if they have access to a domain-adapted machine translation model. Unfortunately, the often limited availability of in-domain parallel data complicates this task. For example, it is difficult to adapt a model to the COVID-19 domain because this issue is too new, and the current available data do not sufficiently cover it.

To alleviate the issue, we propose a method that rapidly adapts a machine translation model to many domains at reasonable costs and time periods with crowdworkers. Fig. 1 shows an overview of our framework. We hypothesize that a small number of in-domain parallel sentences of the target domain are available on the web, and we ask crowdworkers to report these web URLs as a web mining task. Our task does not require translation skills, unlike some previous research (Zaidan and Callison-Burch, 2011; Behnke et al., 2018; Kalimuthu et al., 2019) that attempted manual translations of in-

domain monolingual sentences by crowdworkers. Thus, workers who are not professional translators can participate.

Furthermore, to collect effective parallel sentences, we also vary the crowdworkers' rewards based on the quality of their reported URLs. After collecting parallel sentences by our method, we adapted the machine translation model with the collected, target-domain parallel sentences. Our method has the advantage of being applicable to many domains, in contrast to previous works that use existing parallel/monolingual data.

We experimentally show that our method quickly collects in-domain parallel sentences and improves the translation performance of the target domains in a few days and at a reasonable cost.

Our contributions can be summarized as follows:

- We proposed a new domain-adaptation method that quickly collects in-domain parallel sentences from the web with crowdworkers.
- We empirically showed that crowdworkers are motivated by variable rewards to find more valuable web sites and achieved better performance than under the fixed reward system.

2 Related Work

2.1 Domain Adaptation

Domain adaptation is a method that improves the performance of a machine translation model for a specific domain. The most common method for neural machine translation models is to fine-tune the model with target-domain parallel sentences (Chu and Wang, 2018). Kiyono et al. (2020), who ranked first in the WMT 2020 news shared task (Barrault et al., 2020), fine-tuned a model with a news domain parallel corpus and improved the BLEU scores by +2.2 points. Since the availability of a target-domain parallel corpus is limited, we typically select similar domain sentences from a large parallel corpus (Moore and Lewis, 2010; Axelrod et al., 2011). However, its applicability remains limited because some domains are not covered by existing parallel corpora.

We take a different approach that freshly collects target-domain parallel sentences from the web. Since we do not rely on an existing corpus, our method can be applied to many domains.

2.2 Collecting Parallel Sentences from the Web

Recently, some works successfully built a large-scale parallel corpus by collecting parallel sentences from the web. The BUCC workshop organized shared-tasks of extracting parallel sentences from the web (Sharoff et al., 2015; Zweigenbaum et al., 2017). The ParaCrawl project successfully created a large-scale parallel corpus between English and other European languages by extensively crawling the web (Bañón et al., 2020). Typical bitext-mining projects, including ParaCrawl, took the following steps to identify parallel sentences from the web (Resnik and Smith, 2003): (1) find multilingual websites, which may contain parallel sentences, from the web (Papavassiliou et al., 2018; Bañón et al., 2020); (2) find parallel documents from websites (Thompson and Koehn, 2020; El-Kishky and Guzmán, 2020); (3) extract parallel sentences from parallel web URLs (Thompson and Koehn, 2019; Chousa et al., 2020). Our work focuses on the first step: finding bilingual target-domain web URLs. Bañón et al. (2020) analyzed all of the CommonCrawl data to find crawl candidate websites that contain a certain amount of both source and target language texts. Their method efficiently collected parallel sentences from the web. However, since CommonCrawl only covers a small portion of the web, it may overlook websites that contain valuable resources. Thus, the current web-based corpora (Bañón et al., 2020; Morishita et al., 2020) may not cover all the domains we want to adapt. It is also difficult to focus on a specific topic. In contrast, our work does not rely on CommonCrawl but on crowdworkers who can search the whole web and focus on specific domains.

2.3 Creating Parallel Corpus with Crowdworkers

Some researchers have used crowdsourcing platforms to create new language resources (Roit et al., 2020; Jiang et al., 2018). Some work created a parallel corpus for domain-adaptation by asking crowdworkers to translate in-domain monolingual sentences (Zaidan and Callison-Burch, 2011; Behnke et al., 2018; Kalimuthu et al., 2019). Although this approach is straightforward, it does suffer from several drawbacks. For example, it is often difficult to find a sufficient amount of crowdworkers since translation tasks often require an understanding of both the languages that are actu-

ally being used. Note that although we also use a crowdsourcing platform, our approach entirely differs from the approach introduced in this section, such as asking crowdworkers to do translation tasks.

3 Collecting Parallel URLs with Crowdworkers

Fig. 1 shows an overview of our collecting protocol. Our method asks workers to find URLs that are related to the target domain and written in parallel. We then extract the parallel sentences from these URLs and fine-tune the general-purpose machine translation model with the collected data.

This section is organized as follows: In Section 3.1, we explain why we focus on collecting parallel URLs and describe their advantages. We overview the details of our crowdsourcing task definition in Section 3.2. In Section 3.3, we describe how we extract parallel sentences from the reported URLs. We describe the details of our reward setting in Section 3.4.

3.1 Advantages

Previous works, which adapted a machine translation model to a specific domain, created resources by asking crowdworkers to translate text (Lewis et al., 2011; Anastasopoulos et al., 2020; Zaidan and Callison-Burch, 2011; Behnke et al., 2018; Kalimuthu et al., 2019). In contrast, our method asks workers to find web URLs (instead of translating sentences) that have parallel sentences in the target domain.

This method has two advantages. The first concerns task difficulty. To achieve rapid domain adaptation, the task must be easy enough that many crowdworkers can participate. Thus, we do not assume that the workers fluently understand both the source and target languages. Finding potential web URLs that have parallel sentences is relatively easy and can be done by any crowdworker.

The other advantage involves task efficiency. We asked workers to collect the URLs of parallel web pages instead of parallel sentences because recent previous works successfully extracted parallel sentences from parallel URLs (Bañón et al., 2020). Efficiency is important for our method, since we focus on speed to create a domain-specific model.

3.2 Crowdsourcing Task Definition

We focus on collecting the parallel sentences of languages e and f . We created a web application to accept reports from the crowdworkers and extracted parallel sentences from the reported web URLs. We prepared a development set (a small portion of the parallel sentences) of the target domain and distribute it to the workers as examples of the type of sentences we want them to collect. The crowdworkers are asked to find pairs of web URLs that contain parallel sentences of the target domain. We call this URL pair a parallel URL. Note that we collect the URLs of pages written in parallel; this means that workers act as parallel document aligners. We do not accept parallel URLs that have already been reported by others.

3.3 Parallel Sentence Extraction

After obtaining parallel URLs from workers, we extract parallel sentences from the reported URLs. First, we downloaded the reported web URLs and extracted the texts¹ and removed the sentences that are not in the e or f language based on CLD2². Then we used `vecalign` (Thompson and Koehn, 2019) to extract the parallel sentences, a step that aligns them based on the multi-lingual sentence embeddings LASER (Artetxe and Schwenk, 2019). We discard noisy sentence pairs based on sentence alignment scores³ and do not use them for model training.

3.4 Reward Settings

To bolster the crowdworkers' motivation, reward setting is one of the most important issues (Posch et al., 2019). In this paper, we tested two types of rewards: fixed or variable. In the following, we describe both reward settings.

3.4.1 Fixed Reward

Fixed reward pays a set amount for each reported URL if we can extract at least one parallel sentence from it. This fixed reward setting is one very typical setting for crowdsourcing.

¹Since we expect the workers to act as document aligners, we focus on the reported URLs and do not crawl the links in the reported URLs.

²<https://github.com/CLD20wners/cld2>

³Since `vecalign` outputs a scoring cost where a lower score means better alignment, our implementation removes a sentence pair if its cost exceeds 0.7.

3.4.2 Variable Reward

The key motivation of crowdworkers is probably to earn money (Antin and Shaw, 2012), and thus they try to maximize their earnings (Horton and Chilton, 2010). Since the fixed reward setting only considers the number of reported URLs, workers may report noisy URLs whose texts are not parallel or not in the target domain in an effort to maximize their number of reports.

To alleviate this concern, we tested another reward setting: varying rewards based on the quality of their reported parallel URLs. We hypothesize that the workers will improve their work performance when we pay more for good work and less for poor work.

We defined parallel URLs as those satisfying the following criteria that help improve the translation performance in the target domain: (1) they contain a large number of parallel sentences, (2) the parallel sentences are correctly translated, and (3) the parallel sentences are in the target domain. To reflect these criteria in the reward, we set variable reward r :

$$r = \min(r_{\max}, r_{\min} + \sum_{(x_i, y_i) \in \mathbb{D}} S_a(x_i, y_i) + S_d(x_i)), \quad (1)$$

where \mathbb{D} is a set of parallel sentences extracted from the reported URLs, x_i and y_i are parallel sentences of languages e and f , r_{\min} and r_{\max} are the minimum and maximum reward per report, and $S_a(\cdot)$ and $S_d(\cdot)$ are the sentence alignment and domain similarity scores, which are explained below.

Sentence Alignment Score Suppose n parallel sentences $\mathbb{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$ extracted from the reported URLs. Sentence alignment score S_a is calculated as follows:

$$S_a = \sum_{(x_i, y_i) \in \mathbb{D}} \varsigma(-V(x_i, y_i)), \quad (2)$$

where $V(\cdot)$ is an alignment cost function of `vecalign`, where lower is better, and $\varsigma(\cdot)$ is a sigmoid function that converts the score into the range 0 to 1.

Domain Similarity Score The domain similarity score is based on cross-entropy (Moore and Lewis, 2010):

$$S_d = \sum_{x_i \in \mathbb{D}} \varsigma(H_I(x_i) - H_N(x_i)), \quad (3)$$

where I and N are in-domain and non-domain-specific language models and $H(x_i)$ is the per-word cross-entropy of sentence x_i .

Through our web application, workers can check the results (of their previous reports), which include the reward amounts, the scores, and the number of extracted parallel sentences. These results are available a few minutes after we accept their reports so that they can improve their work and maximize their scores and their payments.

4 Experiments

We carried out experiments to confirm whether different reward settings influenced the workers' performance and translation accuracy. Prior to them, we conducted a preliminary experiment to check the effect of our method in smaller settings. Refer to Section B in the Appendix for this preliminary experiment. In this section, we empirically confirm the effectiveness of our method by focusing on five domains.

4.1 Experimental Settings

In this experiment, we tested English-Japanese translations on five domains: COVID-19, news, science, patents, and legal matters⁴. The details of the domains and the corpus statistics of the development/test sets are shown in Section A.2 in the Appendix. We hired 97 crowdworkers through a crowdsourcing platform called Crowdworks⁵. Each worker was randomly assigned to a single target domain.

We used both the fixed and variable reward setting for the science and patent domains, and only the variable reward setting for the other three domains, since we confirmed that the variable reward setting is effective in the following experiment (see Section 4.2.1). We set the fixed reward at 25 JPY (\simeq 0.23 USD), r_{\min} to 10 JPY (\simeq 0.09 USD), and r_{\max} to 100 yen (\simeq 0.91 USD) for the variable reward⁶. Since our task is much easier than translating sentences, we pay our workers much less than such translators of sentences⁷. Data collection continued for 13 days. We trained

⁴We chose these domains because they require special domain knowledge and are difficult to translate by current models.

⁵<https://crowdworks.jp/>

⁶We paid the workers in JPY since they mainly live in Japan. They are guaranteed at least the minimum wage.

⁷Typically, it requires around 0.15 USD to translate an English word into Japanese.

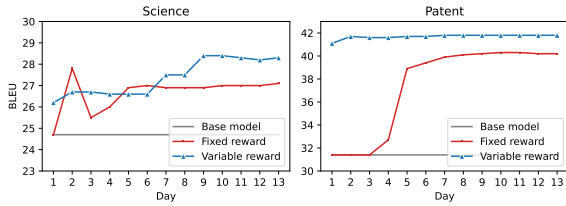


Figure 2: Transition of test set BLEU scores on science and patent domains

in-domain language models with each development set to calculate the domain similarity scores. We used KenLM as an implementation of the n -gram language model (Heafield, 2011) to calculate the domain similarity scores. We trained the in-domain language model with the development set of TICO-19 and the non-domain-specific model with JParaCrawl v2.0 (Morishita et al., 2020).

Translation Model Settings As a neural machine translation model, we employed the Transformer model with its base settings (Vaswani et al., 2017). To train the general-purpose baseline model, we used JParaCrawl v2.0 (Morishita et al., 2020), which contains 10 million English-Japanese parallel sentences and tokenized the training data into subwords with the sentencepiece (Kudo and Richardson, 2018) toolkit. We set the vocabulary size to 32,000 for each language side and removed sentences that exceeded 250 subwords to reduce the noisy sentence pairs.

We trained the baseline model with JParaCrawl until it converged and then fine-tuned it with the newly collected in-domain parallel sentences. See Section A.1 in the Appendix for the detailed hyperparameter settings.

We used SacreBLEU (Post, 2018) to evaluate the translation performance and report the BLEU scores⁸ (Papineni et al., 2002).

4.2 Experimental Results

4.2.1 Fixed or Variable Reward Comparison

First, we address whether the variable reward setting encouraged the workers to find more valuable data. Fig. 2 compares the BLEU scores between the fixed and variable reward settings in the science and patent domains. The variable reward setting achieved higher BLEU scores than the fixed reward setting in both domains. Combined with the pre-

⁸We used NFKC to normalize both the Japanese translations and references since JParaCrawl is normalized by the same procedure.

liminary experiments described in Section B in the Appendix, we conclude that the variable reward setting collects beneficial data. Thus, the following sections mainly discuss the results of the variable reward setting.

4.2.2 Data Collection

Table 1 shows the experimental results on the variable reward setting, including the number of URLs and collected parallel sentences. Our framework collected a large number of parallel sentences for all five domains. The lower half of Fig. 3 shows the transitions of the number of sentences collected with crowdsourcing on the COVID-19, news, and legal domains. For the other domains, see Fig. 6 in the Appendix. The number of collected sentences linearly increased as we continued crowdsourcing.

We carried out the task on the five domains and assigned roughly the same number of workers to each task, but we found that the number of reports differed. This implies that the task’s difficulty might differ depending on the target domain. For example, the science task might be easier than the others because several scientific journals translate abstracts (and make them available on the web) into other languages.

4.2.3 Translation Performance

Table 1 shows the BLEU scores of the baseline and the fine-tuned models with the collected in-domain parallel sentences. The fine-tuned models achieved significantly better accuracy with an average of +7.8 points than the baseline model on all five domains. In particular, our legal domain model improved by +19.7 points. One likely reason is that the legal domain frequently uses words that do not appear in other domains, and the collected in-domain data improved these translations.

The top of Fig. 3 shows the transitions of the BLEU scores as we continued the data collection for the COVID-19, news, and legal domains (see the Base Model and w/Crawled lines). Fig. 6 in the Appendix shows the results of the other domains. All the domains show identical tendencies. Their performance surpassed the baseline on the first or second day of crowdsourcing and continued growing as we collected more data. This supports our assumption that our method can achieve rapid domain adaptation for many domains.

Domain	#URLs	#Sentences	Cost (USD)	Development BLEU			Test BLEU		
				Base model	w/Crawled		Base model	w/Crawled	
COVID-19	6,841	165,838	1,807.7	25.9	28.7	(+2.8)	31.7	34.3	(+2.6)
News	10,712	220,559	2,765.5	19.3	21.2	(+1.9)	20.5	23.1	(+2.6)
Science	10,948	390,303	3,217.8	25.0	27.9	(+2.9)	24.7	28.3	(+3.6)
Patent	4,135	307,104	1,431.3	27.4	36.6	(+9.2)	31.4	41.8	(+10.4)
Legal	5,438	302,747	2,088.0	22.9	42.0	(+19.1)	22.8	42.5	(+19.7)

Table 1: Experimental results for five domains. Model fine-tuned with newly crawled data significantly improved BLEU scores on all of them.

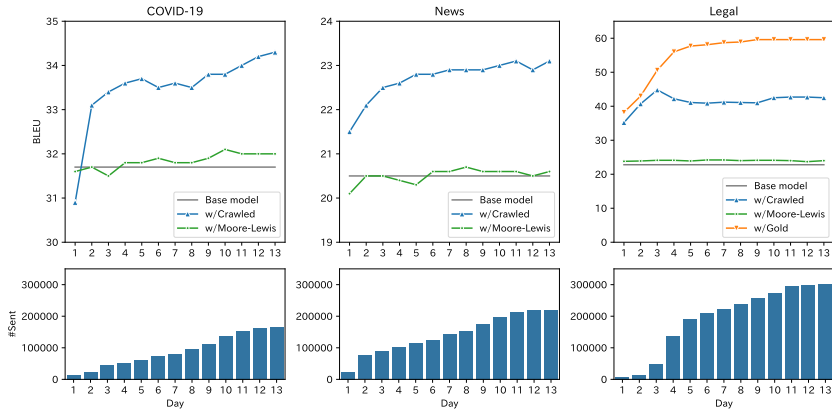


Figure 3: Transition of BLEU scores (top) and sentences collected (bottom) as we continued data collection for the COVID-19, news, and legal domains. Model named w/Moore-Lewis is fine-tuned with domain-relevant sentences extracted from existing general-purpose corpus, as described in Section 4.3. As an upper bound of fine-tuning, we show the scores of the w/Gold model, which was fine-tuned with existing target-domain parallel corpus, as described in Section D.2 in the Appendix.

Domain	Test BLEU			
	Base	w/Crawled		w/ML
COVID-19	31.7	34.3	(+2.6)	32.0 (+0.3)
News	20.5	23.1	(+2.6)	20.6 (+0.1)
Science	24.7	28.3	(+3.6)	25.3 (+0.6)
Patent	31.4	41.8	(+10.4)	32.0 (+0.6)
Legal	22.8	42.5	(+19.7)	24.0 (+1.2)

Table 2: BLEU score comparisons with Moore-Lewis (w/ML)

4.3 Comparison: Selecting In-domain Data from Existing Parallel Corpus

In this section, we compare our method with the existing domain adaption method to answer the following question: Do we really need to collect new data with crowdworkers?

Currently, the most common domain-adaptation method is to find target domain sentences from existing parallel corpora (Chu and Wang, 2018). As with the existing method, we used the one proposed by Moore and Lewis (2010)⁹. We scored all the

⁹Some may be concerned that this method is outdated, but it is still considered a strong domain-adaptation method, since the recent first-ranked system among WMT submissions uses it for selecting relevant data (Junczys-Dowmunt, 2018).

sentences in JParaCrawl and used those considered most relevant to the target domain. We selected the same number of sentences as in our collected data.

Table 2 shows the BLEU scores of each model, and the top of Fig. 3 shows the transition of the BLEU scores (see w/Moore-Lewis). The Moore-Lewis method surpassed the baseline on all five domains, but by a narrow margin. Although their method does not require additional cost, our method achieved significantly better performance with just a small additional cost. Thus the answer to the above question is yes: our method outperformed the existing domain-adaptation method.

5 Conclusion

We introduced a new framework for domain adaptation in machine translation. Our method asks crowdworkers to find parallel URLs related to the target domain. Such a task does not require any professional skills and can be done cheaply by many people. We then fine-tuned the machine translation model with parallel sentences in the target domain extracted from the reported URLs. Through experiments, we empirically confirmed that our

framework significantly improved the translation performance for a target domain within a few days of crowdsourcing and at a reasonable cost. We also confirmed that our variable reward function, which is based on the quality of parallel sentences, changed the behavior of the workers who began to collect more effective parallel sentences, increasing the translation accuracy.

Limitations

We assume that websites containing in-domain parallel sentences are available on the web, which might not be true for some difficult domains. However, since we believe that parallel sentences in neighboring domains are available on the web, we expect our method to improve the translation accuracy on these domains.

We conducted English-Japanese experiments. We expect our method to work on most major language pairs, including German-English and Chinese-Japanese, since there are many parallel websites on these language pairs. However, we haven't yet confirmed whether it does work on very minor language pairs, because finding parallel websites for them is difficult.

Ethics Statement

In the experiments, our crawler strictly followed the "robots.txt" and crawled only from allowed websites. During the experiments, we also ensured that the crowdworkers earned at least the minimum wage.

Acknowledgements

We thank the three anonymous reviewers for their insightful comments.

References

Antonios Anastasopoulos, Alessandro Cattelan, Zi-Yi Dou, Marcello Federico, Christian Federmann, Dmitriy Genzel, Francisco Guzmán, Junjie Hu, Macduff Hughes, Philipp Koehn, Rosie Lazar, Will Lewis, Graham Neubig, Mengmeng Niu, Alp Öktem, Eric Paquin, Grace Tang, and Sylwia Tur. 2020. TICO-19: the translation initiative for COvid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*.

Judd Antin and Aaron Shaw. 2012. Social desirability bias and self-reports of motivation: A study of amazon mechanical turk in the US and India. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2925–2934.

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics (TACL)*, 7:597–610.

Amittai Axelrod, Xiaodong He, and Jianfeng Gao. 2011. Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–362.

Marta Bañón, Pinzhen Chen, Barry Haddow, Kenneth Heafield, Hieu Hoang, Miquel Esplà-Gomis, Mikel L. Forcada, Amir Kamran, Faheem Kirefu, Philipp Koehn, Sergio Ortiz Rojas, Leopoldo Pla Sempere, Gema Ramírez-Sánchez, Elsa Sarrías, Marek Strelec, Brian Thompson, William Waites, Dion Wiggins, and Jaime Zaragoza. 2020. ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4555–4567.

Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. 2020. Findings of the 2020 conference on machine translation (WMT20). In *Proceedings of the 5th Conference on Machine Translation (WMT)*, pages 1–55.

Maximiliana Behnke, Antonio Valerio Miceli Barone, Rico Sennrich, Vilemini Soisoni, Thanasis Naskos, Eirini Takoulidou, Maria Stasimioti, Menno van Zaanen, Sheila Castilho, Federico Gaspari, Panayota Georgakopoulou, Valia Kordoni, Markus Egg, and Katia Lida Kermanidis. 2018. Improving machine translation of educational content via crowdsourcing. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*.

Mara Chinea-Ríos, Álvaro Peris, and Francisco Casacuberta. 2017. Adapting neural machine translation with parallel synthetic data. In *Proceedings of the 2nd Conference on Machine Translation (WMT)*, pages 138–147.

Katsuki Chousa, Masaaki Nagata, and Masaaki Nishino. 2020. SpanAlign: Sentence alignment method based on cross-language span prediction and ILP. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING)*, pages 4750–4761.

Chenhui Chu and Rui Wang. 2018. A survey of domain adaptation for neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, pages 1304–1319.

- Ahmed El-Kishky and Francisco Guzmán. 2020. Massively multilingual document alignment with cross-lingual sentence-mover’s distance. 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 616–625.
- Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita, and Benjamin K. Tsou. 2013. Overview of the patent machine translation task at the NTCIR-10 workshop. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies*, pages 260–286.
- Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the 6th Workshop on Statistical Machine Translation (WMT)*, pages 187–197.
- John Joseph Horton and Lydia B. Chilton. 2010. The labor economics of paid crowdsourcing. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 209–218.
- Youxuan Jiang, Catherine Finegan-Dollak, Jonathan K. Kummerfeld, and Walter Lasecki. 2018. Effective crowdsourcing for a new type of summarization task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 628–633.
- Marcin Junczys-Dowmunt. 2018. Microsoft’s submission to the WMT2018 news translation task: How I learned to stop worrying and love the data. In *Proceedings of the 3rd Conference on Machine Translation (WMT)*, pages 425–430.
- Marimuthu Kalimuthu, Michael Barz, and Daniel Sonntag. 2019. Incremental domain adaptation for neural machine translation in low-resource settings. In *Proceedings of the 4th Arabic Natural Language Processing Workshop*, pages 1–10.
- Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.
- Shun Kiyono, Takumi Ito, Ryuto Konno, Makoto Morishita, and Jun Suzuki. 2020. Tohoku-AIP-NTT at WMT 2020 news translation task. In *Proceedings of the 5th Conference on Machine Translation (WMT)*, pages 145–155.
- Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 66–71.
- William Lewis, Robert Munro, and Stephan Vogel. 2011. Crisis MT: Developing a cookbook for MT in crisis situations. In *Proceedings of the 6th Workshop on Statistical Machine Translation (WMT)*, pages 501–511.
- Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 220–224.
- Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2020. JParaCrawl: A large scale web-based English-Japanese parallel corpus. In *Proceedings of the 12th International Conference on Language Resources and Evaluation (LREC)*, pages 3603–3609.
- Mathias Müller, Annette Rios, and Rico Sennrich. 2020. Domain robustness in neural machine translation. In *Proceedings of the 14th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 151–164.
- Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchi-moto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC)*.
- 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 the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, pages 48–53.
- Vassilis Papavassiliou, Prokopis Prokopidis, and Stelios Piperidis. 2018. Discovering parallel language resources for training MT engines. In *Proceedings of the 11th International Conference on Language Resources and Evaluation (LREC)*.
- Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318.
- Lisa Posch, Arnim Bleier, Clemens M. Lechner, Daniel Danner, Fabian Flöck, and Markus Strohmaier. 2019. Measuring motivations of crowdworkers: The multidimensional crowdworker motivation scale. *ACM Transactions on Social Computing*, 2(2).
- Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the 3rd Conference on Machine Translation (WMT)*, pages 186–191.
- Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics (CL)*, 29(3):349–380.
- Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation.

- In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7008–7013.
- Serge Sharoff, Pierre Zweigenbaum, and Reinhard Rapp. 2015. BUCC shared task: Cross-language document similarity. In *Proceedings of the Eighth Workshop on Building and Using Comparable Corpora*, pages 74–78.
- 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.
- Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.
- Brian Thompson and Philipp Koehn. 2019. Vecalign: Improved sentence alignment in linear time and space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1342–1348.
- Brian Thompson and Philipp Koehn. 2020. Exploiting sentence order in document alignment. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5997–6007.
- Marlies van der Wees, Arianna Bisazza, and Christof Monz. 2017. Dynamic data selection for neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1400–1410.
- 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 31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 6000–6010.
- Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 560–566.
- Omar F. Zaidan and Chris Callison-Burch. 2011. Crowdsourcing translation: Professional quality from non-professionals. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1220–1229.
- Pierre Zweigenbaum, Serge Sharoff, and Reinhard Rapp. 2017. Overview of the second BUCC shared task: Spotting parallel sentences in comparable corpora. In *Proceedings of the 10th Workshop on Building and Using Comparable Corpora*, pages 60–67.

Base model	
Architecture	Transformer (base)
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$) (Kingma and Ba, 2015)
Learning rate schedule	Inverse square root decay
Warmup steps	4,000
Max learning rate	0.001
Dropout	0.3 (Srivastava et al., 2014)
Gradient clipping	1.0
Label smoothing	$\epsilon_{ls} = 0.1$ (Szegedy et al., 2016)
Mini-batch size	320,000 tokens
Updates	24,000 updates
Averaging	Save checkpoint every 200 steps and average the last eight
Implementation	fairseq (Ott et al., 2019)
Parameters	93.2 million
Fine-tuning	
Learning rate	1×10^{-5} (Fixed)
Mini-batch size	32,000 tokens
Updates	8 epochs ¹⁰
Averaging	Save checkpoint every epoch and average the last eight

Table 3: List of hyperparameters

Domain	Development		Test	
	#Sentences	#Tokens	#Sentences	#Tokens
COVID-19	971	21,085	2,100	49,490
News	1,998	45,318	1,000	22,141
Science	1,790	39,377	1,812	39,573
Patent	2,000	60,312	2,300	71,847
Legal	1,313	46,922	1,310	46,842

Table 4: Number of sentences and English tokens in development and test sets

A Detailed Experimental Settings

A.1 Hyperparameters

Table 3 shows the hyperparameter settings used to train a general-purpose machine translation model and fine-tune it with target domain sentences. We did not conduct a hyperparameter search, and almost all the settings were borrowed from previous works (Morishita et al., 2020; Kiyono et al., 2020).

A.2 Datasets

We used TICO-19 (Anastasopoulos et al., 2020) as development and test sets to evaluate the translation performance of the COVID-19 domain. Since the original TICO-19 does not include Japanese translations, professional translators translated the English sentences to create a Japanese reference. We used the development/test sets from the WMT20 news shared task (Barrault et al., 2020) for the news domain and the NTCIR-10 patent translation task for the patent domain. For the science domain, we used ASPEC (Nakazawa et al., 2016), which contains

¹⁰One epoch means the model sees the entire corpus once. Thus the number of updates depends on the data size. We chose this setting because a fixed number of updates has a risk of over-fitting if the fine-tuning data are too small.

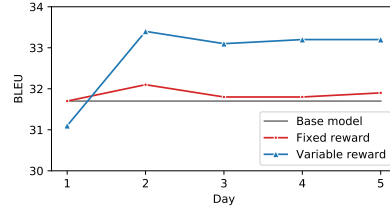


Figure 4: Relationship between BLEU scores and crowdsourcing days for small-scale experiment

excerpts of scientific papers. For the legal domain, we used the Japanese-English legal parallel corpus¹¹. Since it is not divided into development and test sets, we created them by randomly choosing sentences from the entire corpus. The details of the development and test set corpus statistics are shown in Table 4.

B Preliminary Experiments

We carried out a preliminary experiment to determine how the different reward settings influenced the workers’ performance and translation accuracy.

B.1 Experimental Settings

Target Domain and Crowdsourcing Settings In this experiment, we focused on English-Japanese translations in the COVID-19 domain. We assigned ten crowdworkers to each reward setting and asked them to find websites that contained parallel sentences related to the COVID-19 domain. The crowdsourcing continued for five days.

We set the fixed reward at 70 JPY ($\simeq 0.64$ USD) per report. For the variable reward setting, we paid r JPY for each report, as shown by Eq. 1. We set r_{\min} to 20 JPY ($\simeq 0.18$ USD) and r_{\max} to 100 JPY ($\simeq 0.91$ USD). Other model training settings, including the hyperparameters, are identical as in Section 4.1.

B.2 Experimental Results

B.2.1 Data Collection

Table 5 shows the results of crowdsourcing, including the number of reports, extracted parallel sentences, and the payments to the workers. We received almost the same number of reports in both reward settings. However, there was a significant difference in the average number of sentences per report: 10.4 for the fixed rewards and 13.4 for the variable rewards. One likely reason is that the

¹¹<http://www.phontron.com/jaen-law/index.html>

Reward	#URLs	#Sentences	Cost (USD)	Development BLEU		Test BLEU	
				Base model	w/Crawled	Base model	w/Crawled
Fixed	504	5,220	322.8	25.9	26.3 (+0.4)	31.7	31.9 (+0.2)
Variable	503	6,722	284.3		27.1 (+1.2)		33.2 (+1.5)

Table 5: Small-scale experiment’s results (five days of crowdsourcing), including crowdsourcing results and BLEU scores of baseline and model fine-tuned with newly collected in-domain corpus.

workers tried to maximize their rewards. We believe the number of in-domain parallel sentences is one crucial key for improving accuracy, and we reflected this idea in our reward function. Thus it improved the workers’ performance more than the fixed reward setting. With the variable reward setting, we also reduced the cost and obtained even more parallel sentences by reducing the payments to low-quality workers and increasing them to good workers.

B.2.2 Translation Performance

Table 5 shows the BLEU scores of the baseline model and the fine-tuned model with our crawled in-domain parallel data. The model fine-tuned with variable reward data achieved better results than using fixed rewards. We believe the quality of the collected data caused the difference in addition to the number of parallel sentences, as previously mentioned. We compared the domain similarity scores described in Section 3.4.2 to check whether the collected data are related to the target domain and found that the data collected with the variable reward setting achieved higher scores than with the fixed rewards. This implies that the variable reward setting motivated the workers to find parallel web URLs related to the target domain, increasing the accuracy of the fine-tuned model.

Fig. 4 shows how the BLEU scores changed as crowdsourcing continued, and Fig. 5 in the Appendix shows the number of sentences used for this experiment. The fine-tuned model with the variable reward data outperformed the baseline model, even by the second day of crowdsourcing. This result supports our claim that our method helps provide a domain-adapted model in a few days, which is critical in such urgent situations as COVID-19.

From this experiment, we found that a variable reward setting encouraged workers to find more valuable parallel URLs, improved their translation performance in the target domain over a few days, and reduced the cost more than the fixed reward setting.

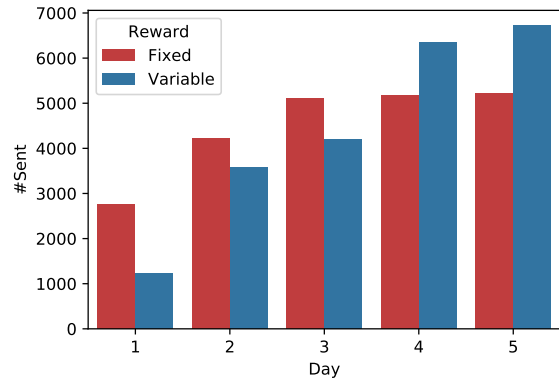


Figure 5: Collected sentences used for fine-tuning in experiment of Fig. 4. See Section B for details.

	Alignment	Domain	Both
Top 20%	34.9	34.2	34.4
Middle 20%	32.9	33.3	33.1
Bottom 20%	30.5	32.3	31.8

Table 6: BLEU scores of model fine-tuned with top/middle/bottom 20% scored sentences on COVID-19 domain test set

C Additional Experimental Results

Fig. 5 shows the numbers of sentences used for fine-tuning in the preliminary experiment (Section B). Fig. 6 shows the transitions of the BLEU scores in the experiment described in Section 4 and the number of sentences collected in the variable reward setting.

D Additional Analysis

D.1 Analysis: Reward Function

We varied the rewards to the workers with the reward function based on the sentence alignment and domain similarity scores. We pondered whether this reward function could correctly measure the data quality. To confirm this, we ordered the collected data with respect to the sentence alignment scores (Eq. 2), the domain similarity scores (Eq. 3), or the sum of both scores. Then we fine-tuned the model with the top/middle/bottom 20% of the sorted data.

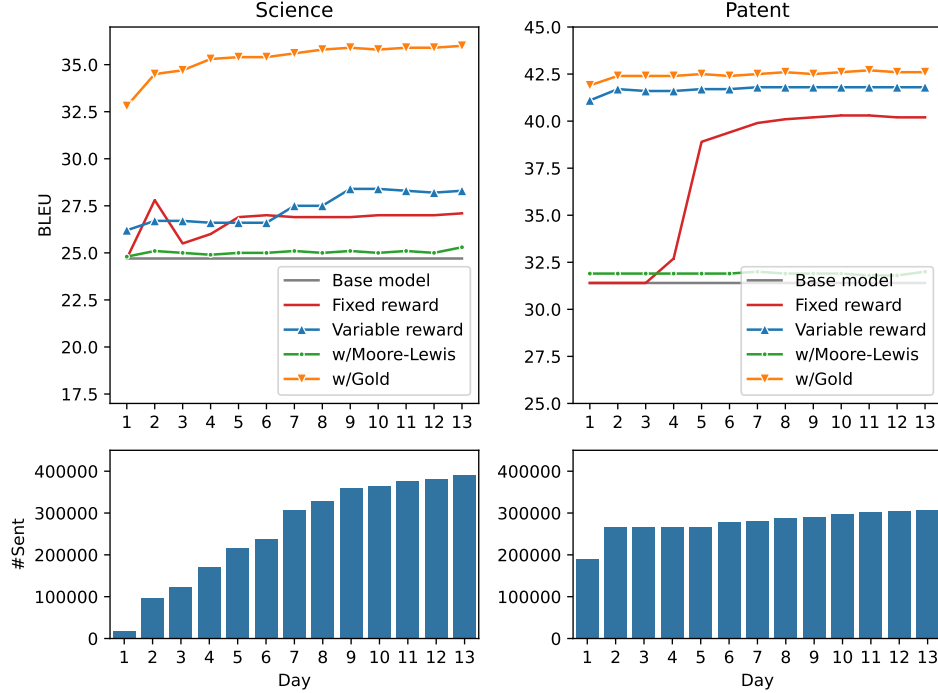


Figure 6: Transitions of BLEU scores (upper-side) and number of collected sentences (lower-side) as we continued data collection for science and patent domains. Detailed explanations can be found in Section 4.

Domain	Test BLEU		
	Base	w/Crawled	w/Gold
Science	24.7	28.3 (+3.6)	36.0 (+11.3)
Patent	31.4	41.8 (+10.4)	42.6 (+11.2)
Legal	22.8	42.5 (+19.7)	59.6 (+36.8)

Table 7: BLEU score comparison with Gold data (w/Gold)

Table 6 shows the BLEU scores of the fine-tuned models on the COVID-19 domain. There is a clear trend that the model fine-tuned with high-scored data achieved higher accuracy, and there is a large gap between the top and the bottom for all the score functions. From this result, we conclude that our reward function correctly measured the quality of the data, and we paid more for high-quality works and less for low-quality works.

D.2 Comparison: Gold In-domain Parallel Corpus

In this section, we compare our collected data with the existing domain-specific parallel corpus. Among the five domains from which we collected sentences, there is a domain-specific parallel corpus for the science, patent, and legal domains. Note that the availability of domain-specific data is quite limited since creating such parallel data requires

professionals, thus incurring heavy costs. Accordingly, this experiment resembles a comparison between our method and the upper bound. In the following, we call this domain-specific parallel corpus the gold data.

As gold data, we used ASPEC (Nakazawa et al., 2016) for the science domain, NTCIR (Goto et al., 2013) for the patent domain, and the Japanese-English legal parallel corpus for the legal domain. For a fair comparison, we randomly selected the same number of sentences as our collected data.

Table 7 shows the BLEU scores of the model fine-tuned with the gold data. Unsurprisingly, the w/Gold models achieved better accuracy than the w/Crawled models. However, the results of some of the latter were close to those of the former, such as the patent domain.

From Fig. 3, we compared the transition of the BLEU scores (see w/Crawled and w/Gold in the legal domain). From the first to third days, our method’s performance resembled that of the w/Gold model. Since room remains for improvements after the fourth day, future work will refine the crowdsourcing protocol.

E Links to Data and Software

E.1 Data

JParaCrawl <https://www.kecl.ntt.co.jp/icl/lirg/jparacrawl/>

TICO-19 <https://tico-19.github.io/>

WMT20 news shared task <http://www.statmt.org/wmt20/translation-task.html>

ASPEC <http://orchid.kuee.kyoto-u.ac.jp/ASPEC/>

NTCIR-10 <http://research.nii.ac.jp/ntcir/permission/ntcir-10/perm-en-PatentMT.html>

Legal parallel corpus <http://www.phontron.com/jaen-law/index.html>

E.2 Software

vecalign <https://github.com/thompsonb/vecalign>

CLD2 <https://github.com/CLD2owners/cld2>

KenLM <https://github.com/kpu/kenlm>

fairseq <https://github.com/pytorch/fairseq>

SacreBLEU <https://github.com/mjpost/sacreBLEU>

sentencepiece <https://github.com/google/sentencepiece>

Biomedical NER for the Enterprise with Distilled BERN2 and the Kazu Framework

Wonjin Yoon^{1†} Richard Jackson^{2†} Elliot Ford² Vladimir Poroshin² Jaewoo Kang^{1,3}

¹Korea University, Seoul, South Korea

²AstraZeneca, Cambridge, United Kingdom

³AIGEN Sciences, Seoul, South Korea

† Equal contribution

{wjyoon, kangj}@korea.ac.kr

{richard.jackson4, elliot.ford, vladimir.poroshin}@astrazeneca.com

Abstract

In order to assist the drug discovery/development process, pharmaceutical companies often apply biomedical NER and linking techniques over internal and public corpora. Decades of study of the field of BioNLP has produced a plethora of algorithms, systems and datasets. However, our experience has been that no single open source system meets all the requirements of a modern pharmaceutical company. In this work, we describe these requirements according to our experience of the industry, and present Kazu, a highly extensible, scalable open source framework designed to support BioNLP for the pharmaceutical sector. Kazu is a built around a computationally efficient version of the BERN2 NER model (TinyBERN2), and subsequently wraps several other BioNLP technologies into one coherent system.

1 Introduction

One of the promises of the applications of A.I. within the pharmaceutical sector is to empower the search for new drugs, and quicken their development into safe, effective medicines. Within the field of NLP, this commonly involves the application of named entity recognition (NER, which is the task of finding entities from a document) and entity linking (EL, also known as grounding, or normalisation), and other techniques to internal and external documents. Documents enhanced with such metadata have a wide variety of use cases, such as improving the performance of enterprise search systems, phase 4 monitoring of adverse events or as a precursor to relationship extraction (for instance in biomedical knowledge graph construction (Geleta et al., 2021)).

Our experience has been that high quality NER remains at the core of many typical NLP use cases within the pharmaceutical industry, and therefore is the prominent focus of our work. BioNER as a field is notable for its technical complexity and

chronic shortage of sufficiently sized training/test datasets, relative to general domain corpora. Although recent advances have produced excellent results on benchmark datasets, recent work (Kim and Kang, 2022) has also suggested that such approaches may be overfit, and may not necessarily generalise sufficiently to meet the needs of a production system.

Similarly, the tendency of academic products to focus on minimising the error rate over a given benchmark ignores practical issues of productionisation, such as the computational complexity of an algorithm, ecological impact and the maintenance of a coherent codebase. While a low overall error rate is undoubtedly important for any enterprise A.I. system, world class performance often comes at the expense of speed. Therefore, striking a balance between an acceptable error rate and other performance metrics is central to user acceptance. We posit that ‘near’ (rather than ‘absolute’) state of the art is sufficient for most use cases.

While several commercial solutions are available to address this requirement, we suggest that a freely available open source solution to deal with the intricacies of this area has not been forthcoming. In this piece, we describe the practical challenges and requirements of enterprise BioNLP analytics. We present our TinyBERN2 biomedical NER model, which utilises weak supervision to address generalisability issues, and our associated Kazu framework by which we deploy it for enterprise applications within a large pharmaceutical company. The Kazu framework and models (including TinyBERN2 and distilled PubMedBERT (Gu et al., 2021)) are open-sourced¹.

¹<https://github.com/AstraZeneca/KAZU>

2 Challenges of BioNLP in the Pharmaceutical Sector and the Kazu Framework

The priorities of academic research in NLP often do not focus on the various practical elements of productionising algorithms within the context of a corporate environment. Nevertheless, this is one of the domains where their outputs can deliver value - via extending existing systems and enabling new projects. The challenges of managing A.I. systems in such environments are acknowledged by the emergence of the field of MLOps, and we repeat some of the most salient aspects relevant to BioNLP here.

2.1 Language/technology agnostic and scalability

The majority of algorithms for BioNLP are typically written in JVM languages or Python, each of which may have dependency conflicts with other algorithms within the overall Kazu pipeline. Here, we utilise the scalable Ray framework (Moritz et al., 2018) which allows different processes to run with distinct Python virtual environments/JVM classpaths, substantially reducing the chance of a conflict.

2.2 Flexibility of datasource ingestion

The biomedical domain is awash with ontologies and knowledgebases, representing various attempts to standardise and model biological concepts. These typically form the basis of EL targets and/or dictionary based NER vocabularies. Thus, we have built a parsing system to allow any data source to be converted into a vocabulary, suitable for curated dictionary based entity matching and/or entity linking.

2.3 Robustness of data model

The biomedical literature is known for the over-representation of certain linguistic phenomena, such as multi section documents/abstracts, nested entities (Alex et al., 2007) and non-contiguous entities (Lever et al., 2020). We note that the data models of many popular NLP frameworks don't contain native support for these concepts, and have thus built these into the standard Kazu dataclasses.

2.4 Extensibility of pipeline design

The current pace of NLP development is extremely rapid. We present Kazu with implementations of

several algorithms that we have chosen based on our preference at the time of writing. However, we recognise that any or all of these are likely to be super-ceded in the short to medium term. Therefore, we have designed Kazu in a modular pipeline fashion, wherein new algorithms can be introduced relatively easily.

2.5 Stability in execution

Executing NLP algorithms over large corpora of text is notoriously unreliable, due to the difficulties in building systems that are able to cope with highly arbitrary input. For instance, certain strings of text may cause NER processes to crash due to high memory usage. Systems that are able to identify problematic text, and either a) avoid processing exceptions or b) recover from such situations are helpful in production environments. To deal with these scenarios, we leverage several techniques such as process memory monitoring/automatic worker restarting, which in turn allows the processing of millions of documents with relative ease.

3 Methods

3.1 Model Architecture

Historically, BioNER approaches have utilised 'sequence tagging' style tasks (Yoon et al., 2019; Lee et al., 2020), wherein a string of text is tokenized, and a model assigns each token a label according to the popular Begin, Inside, Outside schema. However, this single-label classification approach can not properly predict *nested entities* (Katiyar and Cardie, 2018), where the spans of multiple types of entities are overlapped, without additional processing or methods.

To mitigate the problem from nested entity, we applied multi-label label prediction for each token. The method is straight-forward and neither require complicated training strategy nor additional parameters that leads to significant speed reduction in inference.

Our model is composed of BERT layers and a dense output layer. For the case where the number of entity classes is k and using the B-<entity class>, I-<entity class>, and O tag schema (Ramshaw and Marcus, 1999), the output layer o is defined as follows:

$$o = \text{Sigmoid}(hW + b) \quad (1)$$

where $h \in \mathbb{R}^d$ is the output of the final layer of BERT layers for a token $W \in \mathbb{R}^{d*(2*k+1)}$, and

$b \in \mathbb{R}^{2*k+1}$. The output layer will produce a vector (with $2 * k + 1$ elements) of probability for each tokens.

The training objective is to reduce loss between the output of the model (vector of size $2 * k + 1$) and the annotation vector. Each element of the annotation vector represents whether the token is a part of the corresponding entity class. Note that the elements are independent and an annotation vector can label multiple entity classes. The element can be a binary value (hard-label) or a probability value (soft-label) of an off-the-shelf model. We used a standard Binary Cross Entropy loss (as implemented in the pytorch library) as our loss function ².

3.2 Weakly supervised learning

To address the aforementioned generalisability concerns, we adapt a weakly supervised learning strategy (Ratner et al., 2018). Figure 1 shows an example of our training dataset generation. Off-the-shelf models or existing models can be used to generate weakly-labeled datasets. In our case, we utilized predictions of BERN2 (Sung et al., 2022) on PubMed articles, that is available on the official web-page ³ (downloaded Feb 7, 2022 version v1.0). Statistics of the downloaded dataset are shown in Table 1. We pre-processed about 25 Million MEDLINE abstracts available in PubMed website ⁴. In order to reduce ecological footprint during experiments, we used about 10% of BERN2-labeled dataset to make our weakly labeled dataset. Articles in test set of benchmark datasets are filtered out from our weakly-labeled training dataset by PMIDs to prevent any downstream models from directly learning/memorizing of BERN2 predictions.

Some of the obstacles in training with weakly supervised learning are noises and biases added during the labeling (Jiang et al., 2021). In the Section 5.1, we explore both soft-labels (i.e. training with the confidence values of the supervising model) and hard-labels (i.e. training with the categorical labels produced by the supervising model).

3.3 Distillation

Distillation is a key tool to address the scalability requirement as it enables to make computationally

²<https://pytorch.org/docs/1.12/generated/torch.nn.BCELoss.html>

³<http://bern2.korea.ac.kr/>

⁴https://www.nlm.nih.gov/databases/download/pubmed_medline.html

Type	BERN2-labeled	Our weakly-labeled
Abstracts	25,726,681	
Sentences	157,267,033	16,712,485
Words (tokens)	3,646,395,389	438,686,717
Words per sentences	23.18	26.24

Table 1: Statistics of BERN2-labeled dataset and our weakly-labeled dataset. About 10% of the BERN2-labeled dataset is used as the weakly-labeled dataset for the training step of TinyBERN2. *Words* denotes the tokens delimited by spaces or special characters in the sentence.

efficient models while retaining most of the F1 performance. During distillation, both the hidden size and the number of layers are reduced. The former reduces the parameter size, resulting a smaller memory usage, and eventually facilitates inferencing with much larger batchsize. The latter not only reduces memory usage, but also decreases the CPU/GPU time spent for a single example to be processed.

Following the work of Jiao et al. (2020), we applied two-stage approach of distillation. In the first stage, we distillate a biomedical domain specific transformer language model to build a task-independent tiny language model (LM). In the second stage, the distilled LM can be trained for a task-specific dataset (such as an NER task) directly or alternatively, a full-sized transformer model trained on the specific NER task is used as a teacher model, which in turn tunes a task-specific distilled LM.

In our experiment, the teacher model for the first stage is PubMedBERT (Gu et al., 2021). For the second stage, we first train a full sized BERN2 (Sung et al., 2022) on our weakly labeled dataset, which is then used as a teacher model for our final, distilled NER model (TinyBERN2).

4 Experiments and Results

4.1 Benchmark Datasets

For evaluation of our TinyBERN2 model, we chose 8 benchmark datasets of 6 entity classes: Gene/Protein, Disease, Chemical, Species, Cell line, and Cell type (Doğan et al., 2014; Li et al., 2016; Krallinger et al., 2015; Smith et al., 2008; Kim et al., 2004; Gerner et al., 2010; Neves et al., 2013; Kaewphan et al., 2016). While we were designing the experiment, we wanted to examine the generalizability (i.e. ability to predict *unseen entities* that are not in the training dataset (Kim and Kang, 2022), which is essential for real-world use

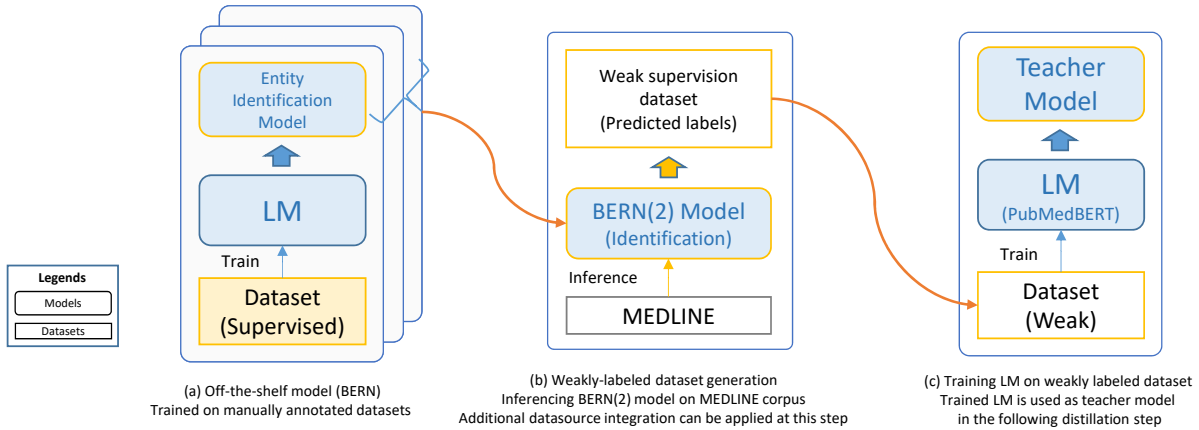


Figure 1: Building a weakly supervised dataset and subsequent training our weakly supervised BERN2 (WS-BERN2) model. We used BERN2 (Sung et al., 2022) to generate weakly-labeled datasets. Alternatively, off-the-shelf models / existing models can be used. Additional datasource integration can be applied at the step (b). Generated weakly-labeled datasets can be used to train a full-sized LM (WS-BERN2, or teacher model in the step (c)) or to train a tiny-sized LM (distilled LM) directly.

cases. To this end, we evaluate our model on several datasets that were not used to train our teacher model (BERN2). These are marked with † in the main experiment table (Table 2).

For the benchmark datasets, we used *MTL-Bioinformatics-2016* GitHub repository⁵ (Crichton et al., 2017) with an additional processing step that ensures all special characters are consistently used as token delimiters. All benchmark datasets were pre-processed to have the same format, where a line contains one token and a corresponding label tag as in CoNLL-X format (Buchholz and Marsi, 2006).

4.2 Results

Table 2 shows our experimental results on benchmark datasets. We compared the performance of the off-the-shelf model (BERN2), weakly-supervised model (WS-BERN2), and distilled weakly-supervised BERN2 model (TinyBERN2) in terms of precision/recall/F1 and computational costs.

4.2.1 Evaluation metrics (accuracy)

We measured entity level Precision, Recall, and F1-score using **SeqEval** library⁶ (Nakayama, 2018). Some datasets contain multiple entity classes. As our model supports multi-label output for each token, we first save model predictions for all types

⁵<https://github.com/cambridgeltl/MTL-Bioinformatics-2016>

⁶<https://huggingface.co/spaces/evaluate-metric/seqeval>

and collect each type separately using the saved output and evaluate entity classes using the collected output. Formally, assume that a model can predict k entity classes and an example has n tokens. If we use BIO-tagging schema the number of labels are $2 * k + 1$ including "O" label. The output of the model can be denoted as a matrix $M \in \mathbb{R}^{(2*k+1) \times n}$. For evaluating i -th ($i \in \{1 \dots k\}$) entity class, we use two rows that marks the given entity class and a row for "O" in M each of them is a vector of length n . These three vectors are merged and form a prediction list, which is used along with the gold standard labels for evaluating an entity class of a benchmark dataset.

In our experiments, F1-scores of our weakly supervised model (WS-BERN2) were analogous to BERN2 model for entity classes where more training data were available (Gene, Disease, Chemical). Our weakly supervised model showed better performance in cross-dataset evaluation (i.e. evaluation on datasets that are not used to train BERN2), which support our assumption on generalisability. As expected, our distilled model, TinyBERN2 showed a lower F1 score across most datasets compared to WS-BERN2, although this was marginal in many cases.

4.2.2 Evaluation metrics (Computational costs)

For evaluation of the models in respect of throughput and speed, we used 26,365 sentences from BC4CHEMD test dataset. One sentence forms a test sample. *Speed* in Table 2 is a measurement for

Group	Benchmark	BERN2 (*)	WS-BERN2	TinyBERN2
		P / R / F1	P / R / F1	P / R / F1
Gene	BC2GM †	82.47% / 80.77% / 81.61%	82.52% / 82.55% / 82.54%	80.80% / 79.37% / 80.08%
	JNLPBA	67.23% / 72.37% / 69.70%	65.88% / 70.14% / 67.95%	64.72% / 69.08% / 66.83%
	CellFinder	63.28% / 52.89% / 57.62%	70.25% / 80.40% / 74.98%	63.67% / 75.71% / 69.17%
Disease	NCBI-disease †	86.33% / 80.94% / 83.55%	87.41% / 88.23% / 87.82%	84.95% / 82.92% / 83.92%
	BC5CDR-Disease	77.14% / 64.76% / 70.41%	78.26% / 67.36% / 72.40%	76.91% / 66.25% / 71.18%
Chemical	BC4CHEMD †	93.66% / 92.06% / 92.86%	91.93% / 91.44% / 91.69%	90.02% / 88.70% / 89.36%
	BC5CDR-Chemical	94.41% / 86.57% / 90.32%	94.55% / 87.59% / 90.94%	94.52% / 86.55% / 90.36%
Cell line	JNLPBA †	50.00% / 76.10% / 60.35%	33.43% / 71.80% / 45.62%	34.85% / 68.80% / 46.27%
	CellFinder	8.52% / 36.07% / 13.78%	10.08% / 45.02% / 16.48%	3.61% / 12.37% / 5.59%
	GELLUS	8.61% / 25.70% / 12.90%	9.01% / 24.02% / 13.11%	7.54% / 18.99% / 10.79%
Cell type	JNLPBA †	60.49% / 67.96% / 64.01%	33.43% / 71.80% / 65.33%	66.41% / 64.06% / 65.22%
	CellFinder	51.12% / 32.19% / 39.50%	48.86% / 31.81% / 38.53%	52.33% / 25.06% / 33.89%
Species	LINNAEUS †	89.13% / 91.56% / 90.33%	80.82% / 44.01% / 56.99%	81.86% / 42.05% / 55.56%
	CellFinder	33.38% / 75.08% / 46.21%	38.03% / 52.96% / 44.27%	46.02% / 50.47% / 48.14%
Model complexity	Backbone model	Bio-LM (Lewis et al., 2020)	PubMedBERT (Gu et al., 2021)	Distilled model
	# Parameters	365M	109M	14M
	# Layers	24	12	4
Throughput (CPU) (batch size = 1)	Speed (s/steps)	0.37 sec/steps	0.043 sec/steps	0.017 sec/steps
	Throughput (samples/s)	2.66 samples/s	22.93 samples/s	57.43 samples/s
Throughput (CPU) (batch size = 32)	Speed (s/steps)	N/A	1.204 sec/steps	0.119 sec/steps
	Throughput (samples/s)	N/A	26.56 samples/s	267.40 samples/s
Throughput (GPU) (batch size = 1)	Speed (s/steps)	0.037 sec/steps	0.012 sec/steps	0.006 sec/steps
	Throughput (samples/s)	26.33 samples/s	83.66 samples/s	160.86 samples/s
Throughput (GPU) (batch size = 32)	Speed (s/steps)	N/A	0.086 sec/steps	0.023 sec/steps
	Throughput (samples/s)	N/A	367.89 samples/s	1354.08 samples/s

Table 2: Performance of our TinyBERN2 model and BERN2 model. Benchmark datasets that are used to train BERN2 is marked with †. *WS-BERN2* denotes our model trained on weakly-supervised dataset. *# Layers* denotes the number of transformers layer in the backbone model, excluding embedding layer and the output layer. *: Performance for the BERN2 may vary with the BERN2 paper (Sung et al., 2022) as we applied different tokenization schema, and application overheads for throughput.

seconds spent for an evaluation step: we divided the number of steps by time spent to process the dataset using mini batch size of 1. The models were loaded in the memory before the experiment.

For BERN2 model (the off-the-shelf model), we installed BERN2 in a local machine to reduce network overheads. The model codes were modified to run without normalization and rule-based post-processing features, with help of the BERN2 authors. For evaluating the memory usage of BERN2, we only report memory usage of `batch_size = 1` setting, as BERN2 doesn’t currently support making batch predictions⁷. For the same reason, we omit throughput results for BERN2.

We used a bare-metal server (Ubuntu Server 16.04.3 LTS) with single NVIDIA TITAN Xp (12GB) GPU and Xeon(R) E5-2630 v4 @ 2.2GHZ (10 Core / 20 Threads) CPU for the experiments.

⁷<https://github.com/dmis-lab/BERN2/issues/10>

5 Discussions

5.1 Effect of training by soft-labeling

In Section 3.2 we suggested that the traditional binary labeling, or hard-labeling, can hinder the weakly-labeled training by adding biases. To examine this hypothesis, we compared the performance of models trained using the hard-label dataset and the soft-label dataset.

Table 3 shows the performance of WS-BERN2 models trained on BIO-tagging with hard-labels and soft-labels (for the comparison between IO-tagging and BIO-tagging, see Section 5.2). In macro average score, WS-BERN2 model trained with soft-label outperformed model trained with hard-label by 0.52%. An interesting observation is that the model benefited from soft-labelling for entity classes with fewer training instances in the weakly-labeled dataset. Here, the macro average improvement was 0.89% in 7 benchmark datasets across the species, cell line and cell type entity

Entity Group	Benchmark	BIO-Hard	BIO-Soft	IO-Soft
		F1	F1	F1
Gene	BC2GM †	82.78%	82.54%	83.76%
	JNLPBA	67.93%	67.95%	68.82%
	CellFinder	73.71%	74.98%	75.36%
Disease	NCBI-disease †	87.38%	87.82%	87.82%
	BC5CDR-Disease	72.67%	72.40%	72.85%
Chemical	BC4CHEMD †	91.81%	91.69%	92.44%
	BC5CDR-Chemical	91.05%	90.94%	91.26%
Cell line	JNLPBA †	44.70%	45.62%	48.27%
	CellFinder	15.66%	16.48%	14.73%
	GELLUS	12.46%	13.11%	12.11%
Cell type	JNLPBA †	64.45%	65.33%	66.60%
	CellFinder	39.27%	38.53%	38.83%
Species	LINNAEUS †	56.70%	56.99%	57.23%
	CellFinder	40.86%	44.27%	43.33%
Macro Average		60.10%	60.62%	60.96%

Table 3: Performance of our weakly-supervised BERN2 models by different labeling schema. Benchmark datasets that are used to train BERN2 is marked with †.

classes. Marginal improvements were observed for entity classes where more training data were available (Gene, Disease, Chemical), with a macro average improvement of 0.14% in 7 benchmark datasets.

5.2 Tagging Schema

Previous works on Biomedical NER tasks have preferred to use the BIO-tagging schema over IO-tagging (i.e. only tag with Inside or Outside), presumably because BIO-tagging can delimit entity spans completely (i.e. IO-tagging cannot delimit two consecutive entities). However, we revisit this convention for the NER task where the model is a transformer based language model and the input text are from the formal scientific literature.

Transformer based language models, such as BERT and BioBERT, use trained tokenizers, such as BPE-tokenizers (Sennrich et al., 2016), and do not remove stopwords or special characters. Instead those tokenizers make stopwords or special characters an independent token.

We hypothesized that in scientific literature, the writer of the text tend to express themselves in a way that causes entities to be wrapped with non-entities, stopwords (such as *and*, *or*) or punctuation characters (such as ",", ".", ":", "(", and "''").

From this perspective, we conducted a preliminary study using BERN2 predictions on 106,921 sentences from randomly sampled articles. Across 102,569 entities, 99,814 entities (97.3%) do not have adjacent entities, in the sense that an entity

span starts right after an entity span of the same class ends - only 466 entities were found to have adjacent entities that cannot be delimited with the IO-tagging schema.

... metabolism of goldfish Carassius auratus (L.).

Figure 2: An example of adjacent entities (*goldfish* and *Carassius auratus*).

Based on this observation, we conducted an experiment to evaluate the usability of IO-tagging. Performances of WS-BERN2 models trained on IO-soft and BIO-soft are denoted in Table 3. Based on the macro average score, the model using IO-tagging outperformed the model with BIO-schema. For the performance on entity classes across the benchmark datasets, 11 scores output 14 were improved by using IO-tagging. Using IO-tagging was also beneficial for the aspect of the computational cost required for the training convergence (i.e. fully trained), as the model is less complex.

However, we do not recommend the IO-tagging model for enterprise use cases, as the input samples cannot be guaranteed to be restricted in the scientific/academic writing with complete punctuation. Absence of statistics for manually-labeled dataset remains as limitation of this auxiliary study and is a topic for further research.

5.3 Enterprise usage of Kazu

We have integrated our TinyBERN2 model into our new Kazu framework, alongside other components for abbreviation expansion (Neumann et al., 2019), entity linking (Liu et al., 2020) and additional novel algorithms outside the scope of this paper (full details of the other Kazu components can be found at ⁸).

Kazu is deployed at AstraZeneca enterprise wide, and is already in use as a core component for projects such as biological knowledge graph (BIKG) construction (Geleta et al., 2021) and clinical trial design via enabling the structured search of clinical studies. An execution over PubMed (abstracts) and PubMed Central (full text documents) has extracted the following (uniquely mapped) references: 22 532 diseases, 19 884 genes, 18 715 drugs, 6469 anatomy references, 5 372 cell line references, and 53 cell type references. Additional deployments of Kazu for other internal projects are planned in the near future.

⁸<https://github.com/AstraZeneca/KAZU>

Limitations

One of our objectives in this work was to establish that the TinyBERN2 model is highly performant, both in terms of accuracy metrics and computational efficiency. However, we were limited to testing this on just a single type of CPU and GPU. Since hardware varies dramatically, it is difficult to predict the precise throughput gains for any single setup. In addition, BERN2 incorporates several elements of pre/post processing, that it wasn't possible to completely disable during throughput testing, which will impact the reported throughput and accuracy results to a small degree. Nevertheless, we expect our conclusions to be broadly consistent, given the massive reduction in parameters/layers of the TinyBERN2 model vs the original BERN2.

Notably, both the full BERN2 and our weakly supervised/distilled versions struggled to achieve high F1 on the cell line/type classes. This may be due to inconsistencies in the annotation schema of the cell line/type datasets. However, we also suspect that this is due to the fact our model does not make use of dictionary features. This is consistent with the observations of Kaewphan et al (Kaewphan et al., 2016), in that ML models without dictionary support tend to perform poorly on this entity class. Further work will seek to supplement our approach to weak supervision with such dictionary features for entity classes that are likely to benefit.

Regarding the sizing of our TinyBERN2 model, we followed the recommendations of the original TinyBERT paper, and did not attempt a hyperparameter search to find the optimal trade off between throughput and performance degradation. Further work should explore this aspect.

Ethics Statement

This work complies with the ACL Ethics Policy.

Acknowledgements

We would like to express our gratitude to Antoine Lain (University of Edinburgh) for helping authors to collect and unify the format of benchmark datasets and Mujeen Sung and Minbyul Jeong for providing NER predictions and information for the inference speed experiments. This work is partially funded by National Research Foundation of Korea [NRF-2020R1A2C3010638], the Korea Health Industry Development Institute (KHIDI)

[HR20C0021] and ICT Creative Consilience program [IITP-2022-2020-0-01819] funded by Government of Republic of Korea. We would also like to thank Rolando Fernandez for his development work on Kazu.

References

- Beatrice Alex, Barry Haddow, and Claire Grover. 2007. Recognising nested named entities in biomedical text. In *BioNLP@ACL*.
- Sabine Buchholz and Erwin Marsi. 2006. [CoNLL-X shared task on multilingual dependency parsing](#). In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York City. Association for Computational Linguistics.
- Gamal Crichton, Sampo Pyysalo, Billy Chiu, and Anna Korhonen. 2017. A neural network multi-task learning approach to biomedical named entity recognition. *BMC bioinformatics*, 18(1):1–14.
- Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.
- David Geleta, Andriy Nikolov, Gavin Edwards, Anna Gogleva, Richard Jackson, Erik Jansson, Andrej Lamov, Sebastian Nilsson, Marina Pettersson, Vladimir Poroshin, et al. 2021. Biological insights knowledge graph: an integrated knowledge graph to support drug development. *Biorxiv*.
- Martin Gerner, Goran Nenadic, and Casey M Bergman. 2010. Linnaeus: a species name identification system for biomedical literature. *BMC bioinformatics*, 11(1):85.
- Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.
- Haoming Jiang, Danqing Zhang, Tianyu Cao, Bing Yin, and Tuo Zhao. 2021. [Named entity recognition with small strongly labeled and large weakly labeled data](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1775–1789, Online. Association for Computational Linguistics.
- 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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

- Suwisa Kaewphan, Sofie Van Landeghem, Tomoko Ohta, Yves Van de Peer, Filip Ginter, and Sampo Pyysalo. 2016. Cell line name recognition in support of the identification of synthetic lethality in cancer from text. *Bioinformatics*, 32:276 – 282.
- Arzoo Katiyar and Claire Cardie. 2018. [Nested named entity recognition revisited](#). 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 861–871, New Orleans, Louisiana. Association for Computational Linguistics.
- Hyunjae Kim and Jaewoo Kang. 2022. How do your biomedical named entity recognition models generalize to novel entities? *Ieee Access*, 10:31513–31523.
- Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics.
- Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. 2015. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7.
- 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.
- Jake Lever, Russ B. Altman, and Jin-Dong Kim. 2020. Extending textae for annotation of non-contiguous entities. *Genomics & Informatics*, 18.
- Patrick Lewis, Myle Ott, Jingfei Du, and Veselin Stoyanov. 2020. [Pretrained language models for biomedical and clinical tasks: Understanding and extending the state-of-the-art](#). In *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 146–157, Online. Association for Computational Linguistics.
- Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wieggers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database: The Journal of Biological Databases & Curation*, 2016.
- Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2020. [Self-alignment pre-training for biomedical entity representations](#). *CoRR*, abs/2010.11784.
- Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica. 2018. [Ray: A distributed framework for emerging AI applications](#). In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, Carlsbad, CA. USENIX Association.
- Hiroki Nakayama. 2018. [seqeval: A python framework for sequence labeling evaluation](#). Software available from <https://github.com/chakki-works/seqeval>.
- 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.
- Mariana Neves, Alexander Damaschun, Nancy Mah, Fritz Lekschas, Stefanie Seltsmann, Harald Stachelscheid, Jean-Fred Fontaine, Andreas Kurtz, and Ulf Leser. 2013. Preliminary evaluation of the cellfinder literature curation pipeline for gene expression in kidney cells and anatomical parts. *Database*, 2013.
- L. A. Ramshaw and M. P. Marcus. 1999. [Text Chunking Using Transformation-Based Learning](#), pages 157–176. Springer Netherlands, Dordrecht.
- Alexander J. Ratner, Braden Hancock, Jared A. Dunmon, Roger E. Goldman, and Christopher Ré. 2018. Snorkel metal: Weak supervision for multi-task learning. *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*.
- 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.
- Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. 2008. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):S2.
- Mujeen Sung, Minbyul Jeong, Yonghwa Choi, Donghyeon Kim, Jinhyuk Lee, and Jaewoo Kang. 2022. [Bern2: an advanced neural biomedical named entity recognition and normalization tool](#). *arXiv preprint arXiv:2201.02080*.
- Wonjin Yoon, Chan Ho So, Jinhyuk Lee, and Jaewoo Kang. 2019. Collabonet: collaboration of deep neural networks for biomedical named entity recognition. *BMC bioinformatics*, 20(10):55–65.

Large-scale Machine Translation for Indian Languages in E-commerce under Low Resource Constraints

Amey Patil, Nikesh Garera

Flipkart

{amey.patil,nikesh.garera}@flipkart.com

Abstract

The democratization of e-commerce platforms has moved an increasingly diversified Indian user base to shop online. We have deployed reliable and precise large-scale Machine Translation systems for several Indian regional languages in this work. Building such systems is a challenge because of the low-resource nature of the Indian languages. We develop a structured model development pipeline as a closed feedback loop with external manual feedback through an Active Learning component. We show strong synthetic parallel data generation capability and consistent improvements to the model over iterations. Starting with 1.2M parallel pairs for English-Hindi we have compiled a corpus with **400M+** synthetic high quality parallel pairs across different domains. Further, we need colloquial translations to preserve the intent and friendliness of English content in regional languages, and make it easier to understand for our users. We perform robust and effective domain adaptation steps to achieve colloquial such translations. Over iterations, we show **9.02** BLEU points improvement for English to Hindi translation model. Along with Hindi, we show that the overall approach and best practices extends well to other Indian languages, resulting in deployment of our models across 7 Indian Languages.

1 Introduction

As one of the largest e-commerce platform, we support a very diverse user base in terms of regional languages. Product Descriptions, Catalog Attributes, and Product Reviews help customers understand and compare various products available on the platform. For the growing user-base in India with non-English background, providing this information in regional Indian languages makes their shopping experience more informative and friendly. With only 10% of the Indian population

being versed in English¹, vernacular support is vital for the platform and its diverse users. In this work, we develop Machine Translation System to translate the available product data from English to regional languages to address this problem. Given the size of the Product Catalog and user base, the volume of the data to be translated is in the order of 100s of millions. This poses a challenge to build Translation Systems that are robust, reliable, and precise at scale.

The low resource nature of Indian Languages² is another challenge for data-hungry deep networks such as Transformer(Vaswani et al., 2017). Given a large enough parallel corpus, the Transformer model can learn the inter-lingual mappings very well, even for very long sequences. These models can generate human-level precision translations for some resource-rich European languages(Popel et al., 2020). So theoretically, if we can get a large enough parallel corpus for Indian languages, we can solve the Automatic Machine Translation for Indian languages.

We build a training pipeline that can take monolingual corpus(abundantly available from public and in-house sources) and generate a high-quality synthetic parallel corpus. This is an efficient and effective approach, especially when paired with the Active Learning component over model iterations. For Hindi, starting with 1.2M parallel examples, we have compiled over 400M synthetic parallel examples with numerous model iterations.

Translation is an inherently one-to-many task where a single text can have various correct translations. The domain gap between the e-commerce domain and public domain (news, government sites, Wikipedia, books, etc.) is significant. To showcase this, Figure 1 has colloquial and non-colloquial

¹https://en.wikipedia.org/wiki/2011_Census_of_India#Language_demographics

²The most extensive parallel corpus has 8.56M English-Hindi translation pairs from Samanantar Dataset(Ramesh et al., 2021)

English input	Colloquial Translation	Non-Colloquial Translation
A good quality product for you	आपके लिए एक अच्छी क्वालिटी का प्रोडक्ट	आपके लिए एक अच्छी गुणवत्ता का उत्पाद
It is perfect for active wear, road ripping or for casual day out.	यह एक्टिव वियर, रोड रिपिंग या कैजुअल डे आउट के लिए परफेक्ट है।	यह सक्रिय पहनने, सड़क पर दौड़ने या आकस्मिक दिन के लिए एकदम सही है।
Decent product at this price segment	इस प्राइस सेगमेंट में अच्छा प्रोडक्ट है।	इस कीमत वर्ग में बेहतरीन उत्पाद।
And to Filter Ultraviolet Rays.	और अल्ट्रावायलेट रे को फिल्टर करने के लिए है।	और पराबैंगनी किरणों को छानने के लिए।
Q: Sound quality is how?	प्रश्न: साउंड क्वालिटी कैसी है?	प्रश्न: ध्वनि की गुणवत्ता है कैसे?
clarity is just awesome in it.	क्लैरिटी बस कमाल है इसमें।	इसमें स्पष्टता बस कमाल है।

Figure 1: Both colloquial and non-colloquial translations are correct, but for E-commerce platform we need more colloquial translation styles.

Hindi translations for a source sentence in English. Both of these translations are correct, but as an e-commerce platform, we refrain from using non-colloquial and infrequently used words as it decreases the appeal of the information from the colloquial e-commerce English domain.

Based on the final training steps, translation models can generate appropriate translations at inference. We fine-tune the model only using the colloquial in-domain data with robust domain adaptation steps to get more colloquial translations.

Our contributions in this paper are as follows:

- **Synthetic Parallel Corpus Generation:** With the help of sub-modules, we generate a vast amount of high-quality parallel corpus solving for low-resource Indian Languages.
- **Iterative Model Training Pipeline:** With the help of data cleaning and filtering modules, we showcase how we iteratively improve the Translation models significantly with Active Learning steps.
- **Large-Scale High Precision and Colloquial Models:** Finally, we provide large-scale Machine Translation models with high precision and domain-adapted colloquial styles for several Indian Languages.

2 Related Work

Transformers (Vaswani et al., 2017) are widely used architecture for seq2seq tasks. Along with Unigram-based subword tokens, the fully attention-based model performs very well for Translation tasks, even for longer sequences. Translation is a well-explored area, and even for low-resource settings, significant work has already been done. Along the lines of data gathering - collecting parallel corpora (Ramesh et al., 2021), mining multilingual sets and retrieving parallel entries (Tran et al.,

2020), iterative cross-lingual alignments (Philip et al., 2021) has been explored. Zhang et al. (2020), showed parallel corpus filtering on web crawled data.

Transfer Learning is also a convenient approach to improve final model performance in low resource settings. (Rothe et al., 2020) explored leveraging large language models trained on unlabelled data for translation tasks. This approach works well only if we have strong pre-trained models. For Indian language settings, this is typically not the case. Also, synthetic data generation is very inefficient without active learning. (Imankulova et al., 2019) shows that translation models can help with pseudo labeling, but this improvement saturates without external feedback. (Peris and Casacuberta, 2018) has explored an active learning framework for machine translation. Gupta et al. (2021) investigate the active learning methods for Machine Translation in Indian Languages settings. Lample et al. (2017) even shows that completely unsupervised Machine Translation is possible using just monolingual data. But these practices don't work in large-scale settings. Given a large amount of good quality parallel data, supervised methods still beat other weak methods. Especially for production settings, there has not been much exploration done at large-scale systems starting with low resource settings.

3 Overall Pipeline

We use Transformer encoder-decoder model with 6 encoder layers and 6 decoder layers with hidden size of 512. We use 32,000 unigram subword tokens trained on data from all domains. This configuration has 93M parameters. As a pre-processing step, we split long paragraphs into sentences and translate independent sentences using the Transformer model.

3.1 Datasets Used

We use all publicly available parallel corpus from various domains within commercial licensing restrictions. Also, we conduct internal operations for parallel corpus creation for in-domain sampled datasets from the Product Descriptions, Catalog Attributes, Search, and Product Reviews. This operation is costly and is only done with the Active Learning step. Apart from parallel corpus, our pipeline heavily relies on synthetic data generation, for which we use publicly available monolingual corpus from general domain compiled from various sources (Wenzek et al., 2020; Abadji et al., 2022; Barrault et al., 2019). The details for the datasets are listed in table 1.

Language	Public Parallel Corpus	In-House Tagged Corpus	Public Mono. Corpus
Hindi	0.89M	0.28M	167M
Tamil	0.86M	0.46M	80M
Telugu	0.32M	0.44M	23M
Bengali	2.13M	0.44M	79M
Marathi	0.448M	0.16M	15M
Malayalam	0.61M	0.16M	32M
Kannada	0.05M	0.20M	18M

Table 1: Monolingual Datasets used in Synthetic Data Generation along with Parallel Corpus

3.2 Monolingual Data Processing

To generate synthetic parallel corpus, we use well-known Back-Translation methods (Sennrich et al., 2016) to translate Indic monolingual corpus back to English. The corpus we use from the public domain is already curated and cleaned. So cleaning Indic monolingual data is easy with some basic text-normalization, rare character filtering, punctuation fixes, etc.

Apart from back-translations, we also use Forward Translations, where we translate monolingual source text to the target language with an imperfect translation model. Forward Translations are a crucial part of our training pipeline. But the quality of synthetic pairs heavily impacts the final model training. Domains such as English Reviews and Search queries can be very noisy with spelling errors, punctuation errors, case errors, etc. While generating translations for these noisy entries, the quality of the translations is limited by the

noisy input itself. Hence before using monolingual data for synthetic parallel data generation, we filter out unclean English texts from the corpus using the pipeline as shown in Fig. 2. We use BERT based classifier model to detect noisy texts from the monolingual corpus. To improve the Translation model robustness, we (1) Correct some of the noisy data filtered out from monolingual corpus to get translations even for noisy text inputs and (2) introduce noise to already clean input texts. We use In-House Transformer-based Encoder-Decoder Spell Correction models to correct unclean texts for search queries and reviews. And as spell correct models have low precision (benchmarks detailed in table 2) we again filter out unclean data from spell corrected set as shown in Fig. 2. We add pairs $\langle \text{noisy text}, \text{translation from cleaned corrected text} \rangle$ as the translation pairs in generated training data.

Data Stream	General Domain	Search	Reviews
F1 score - Noisy text classification	95.03	90.24	92.55
Spell Correct Rate	-	80.53%	55.75%

Table 2: Monolingual Data Cleaning. (Spell Correction rate is the percent of unclean text model corrects properly)

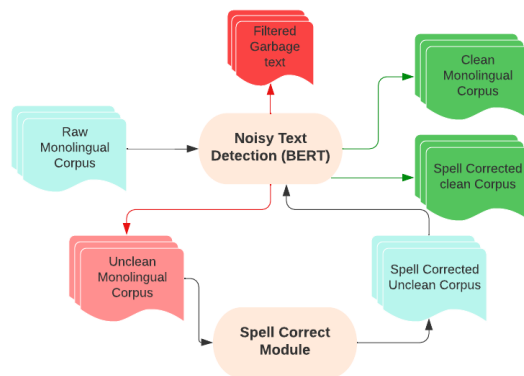


Figure 2: Monolingual Data Cleaning and Spell Correction pipeline.

3.3 Translation Quality Estimation

We monitor and filter imperfect parallel pairs with two methods:

- **Translation model Uncertainty score:** The transformer model uses predictions from a

softmax layer to generate each token output. The output of this layer is the probability distribution over the vocabulary for each token. When the probability of the predicted token is low - the model is more uncertain about the token prediction and vice versa. We aggregate this metric over the entire output sequence and normalize it for the output length to get a final uncertainty score for a translation.

- **Independent BERT based Quality estimation:** Given a source and target sequence, we train a multilingual BERT (Devlin et al., 2018) based classifier, which predicts if the sequences are perfect parallel pairs. The BERT-based classifier model is trained on a set of correct translation pairs (pooled from available high-quality manual translation pairs) and noise-induced pairs from the correct translation pairs with multiple levels of translation errors. To get the final translation score, we pass both the source-target and target-source combination of pairs to a pre-trained BERT encoder and use concatenated context for the classification head.

$$quality_score(x_{1..T}, y_{1..T'}) = h([B(x_{1..T}, y_{1..T'}), B(y_{1..T'}, x_{1..T})]) (1)$$

Model	Precision (good trans.)	Recall (bad trans.)	Overall F1 score
Uncertainty Scoring	0.8889	0.8375	0.8554
BERT Translation Scoring	0.8091	0.6750	0.8166
Ensemble	0.8899	0.8500	0.8264

Table 3: Translation Quality Estimation Benchmark.

As the Translation model Uncertainty score can still be biased toward the erroneously predicted tokens, the independent translation scoring is a good supplement for data filtering. We use an ensemble of two translation quality estimation methods and reject translations setting up high rejection recall. The evaluation scores for both models and ensemble are detailed in Table 3. The final filtered data counts are detailed in table 4 for Hindi language. As expected, rejected synthetic translations are very high for search and reviews set as the stream has very noisy inputs.

Monolingual Dataset	Data Language	Dataset Size	Final Filtered Syn. Parallel Corpus Size
CC-100	Hindi	94.08M	83.09M
OSCAR	Hindi	12.85M	10.77M
news-crawl	Hindi	48.86M	45.11M
Wikipedia	Hindi	1.85M	1.61M
Our Product Descriptions	English	71.32M	70.45M
Our Catalog Attributes	English	64.82M	56.73M
Our Reviews	English	65.80M	44.79M
Our Search	Hindi	29.81M	29.14M
Our Search	English	218.71M	83.03M
Total	-	608.1M	424.72M

Table 4: Monolingual datasets used and back-translated or forward translated dataset size and filtered synthetic corpus size.

3.4 Pipeline with Active Learning

To generate the synthetic parallel corpus and train Translation models using this corpus, we use a pipeline demonstrated in figure 3 in an iterative manner. The detailed algorithm is mentioned in Algo. 1.

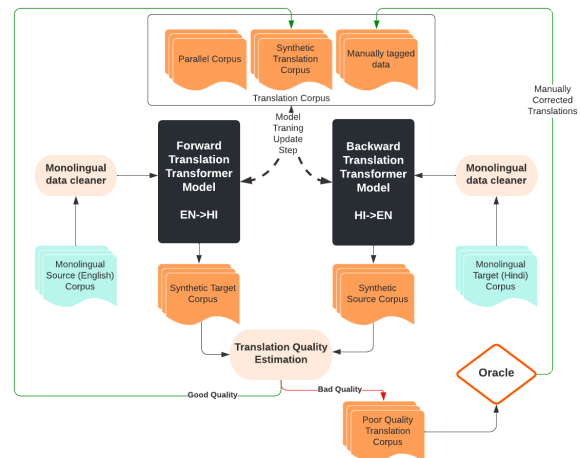


Figure 3: Model Training and Synthetic Data Generation pipeline

We start with English to Indic and Indic to English translation models trained with publicly available and in-house parallel corpus. This base corpus provides good first-version translation models for our iterative pipeline. Iteratively, we process more monolingual data through the pipeline and add good quality synthetic corpus to the training set. The monolingual in-domain text, which the model can not translate accurately, detected by the

Algorithm 1: Training + Data generation pipeline.

Models : M_f (Forward Translation model)
 M_b (Backward Translation model)
 M_n (Noisy text detection BERT)
 M_s (Spell correct model)
 M_q (Translation Quality Est.)

Data : P (Existing parallel corpus)
 C_s (mono. source lang. corpus)
 C_t (mono. target lang. corpus)

```

1 begin
2    $M_f = \text{TransformerTraining}(P)$ 
3    $M_b = \text{TransformerTraining}(P)$ 
4   repeat
5      $C_{s\_clean}, C_{s\_noisy} = M_n(C_s)$ 
6      $C_{s\_corr} = M_{spell}(C_{s\_noisy})$ 
7      $C_{s\_corr\_clean}, C_{s\_corr\_noisy} =$ 
8        $M_n(C_{s\_corr})$ 
9      $C'_{s\_clean} = \text{Translate}(C_{s\_clean}, M_f)$ 
10     $C'_{s\_corr\_clean} = \text{Translate}(C_{s\_corr\_clean},$ 
11       $M_f)$ 
12     $C'_t = \text{Translate}(C_t, M_b)$ 
13     $S = (C_{s\_clean}, C'_{s\_clean}) + (C_{s\_corr\_clean},$ 
14       $C'_{s\_corr\_clean}) + (C'_t, C_t)$ 
15     $S_{good}, S_{poor} = M_{quality}(S)$ 
16     $S_{s\_poor} = \text{sample}(S_{poor})$ 
17     $S_{corr} = \text{oracle}(S_{s\_poor})$ 
18     $TR = S_{good} + S_{corr} + P$ 
19     $M_f = \text{TransformerTraining}(TR)$ 
20     $M_b = \text{TransformerTraining}(TR)$ 
21     $C_s = \text{collect}()$ 
22     $C_t = \text{collect}()$ 
23  until Satisfactory precision achieved;
24 end

```

Translation Quality Estimation module, is pooled, and a diverse batch is sampled from this set to get corrected by manual annotators. This batched Active Learning is crucial in the iteration and makes the forward translations feasible. While re-training the model in the next model iteration, we have filtered good synthetic translations generated by the model and manual translations instead of imperfect translations the model produces. This is an overall translation corpus quality update; hence we train improved Translation models in each iteration.

3.5 Domain Adaptation

As we need colloquial translations in the output, we have to fine-tune the pre-trained models on all domain corpus using just the in-domain colloquial dataset. As evident from Table 5, BLEU scores jump sharply when the model is fine-tuned on the in-domain small training set. This shows that domain gap with general domain and e-commerce colloquial domain is significant. In-domain Forward Translations(forward translated in-domain monolingual corpus) are crucial in this step as the cleaned

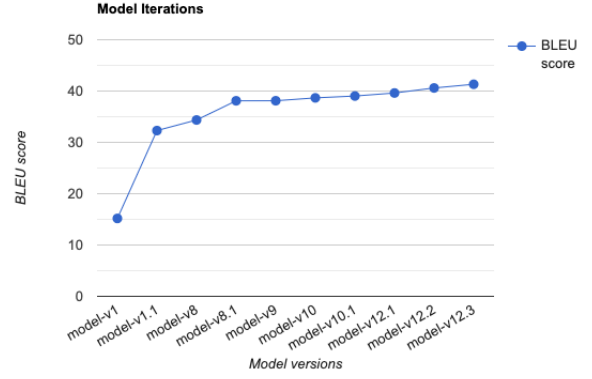


Figure 4: Snapshot of selected models. Iterations vs Product Description BLEU scores for English-Hindi Translation.

Model	Training Steps & Corpus	PD BLEU
Google	-	36.27
Azure	-	29.29
IndicTrans	Samanantar Dataset	31.15
model-v1	Public parallel corpus	15.18
model-v1.1	v1 =>In-domain fine-tuning	32.3
model-v2	Public Parallel Corpus + 50M back-translations	32.3
model-v2.1	v2 =>In-domain fine-tuning	37.59
model-v8	Public Parallel Corpus + 150M back-translations	34.36
model-v8.1	v8 =>In-domain fine-tuning	38.1
model-v9	v8 =>Forward translations	38.11
model-v10	v8 =>Filtered Forward translations	38.56
model-v12	Public Parallel Corpus + 150M back-translations + Filtered Forward translations	37.22
model-v12.1	v12 =>In-domain fine-tuning	39.62
model-v12.2	v12 =>Active Learning (+50k)	40.6
model-v12.3	v12 =>Active Learning (+80k)	41.32

Table 5: Hindi Product Description BLEU scores

and filtered high-quality forward translations help bridge the domain gap and provide much more capable pre-trained models. This ensures that the model does not go through over-fitting or catastrophic forgetting(for the in-domain set), and we get a more robust and reliable model at scale. Table 1 has some examples where our model produces more colloquial translations and refrains from using non-colloquial and non-friendly Hindi words.

3.6 Model Iterations

As evident from Table 4, the BLEU scores are drastically improved in each synthetic data addition step. The best model is improved by **+9.01** BLEU scores over the v1.1 model which does not use any synthetic corpus. The size of back-translated cor-

Language	EN->X PD		EN->X WAT21		X->EN WAT21		Our Translation Accuracy	
	Ours	Google	Ours	Google	Ours	Google	PD	Catalog Attributes
Hindi	41.32	36.27	32.95	32.5	37.85	36.7	95.76%	97.39%
Tamil	44.83	31.86	10.65	8.98	25.37	23.51	94.36%	95.54%
Telugu	39.69	30.78	4.34	4.21	26.28	25.66	90.87%	94.31%
Bengali	30.65	24.33	7.56	5.05	22.51	20.52	98.87%	91.13%
Marathi	37.37	28.86	12.96	12.6	28.07	26	82.05%	95.14%
Kannada	31.32	24.19	12.85	12.9	29.61	24.75	90.38%	96.94%
Malyalam	30.57	27.83	5.09	10.6	28.32	27.2	-	93.32%

Table 6: BLEU scores comparing best public API and Manual Translation Accuracy for our Product Descriptions(PD) and Catalog Attributes.

pus is also impactful even in the range of 10s of million entries, as more data helps significantly. Forward translations are very critical part of the synthetic corpus, as theoretically the quality of forward translations is limited by the performance of the translation model itself used to generate the forward translations. This is where translation quality estimation plays a crucial role for filtering out low quality translations. From Table 4, the model v9 performs very similar to v8.1, which is used to generate forward translations for a large set. But once we filter out imperfect translations, even forward translations show an improved final v10 model. Finally, the additional small set of manual translations generated from Active Learning step over these imperfect translations provides even better v12.* models.

Hindi Model	Test set	Good Trans.	Can be better Trans.	Bad Trans.
Catalog, PD Model	PD	53%	42%	5%
Google	PD	14%	51%	35%

Table 7: English to Hindi Translation evaluation for Product Descriptions(PD)

4 Results and Discussion

We benchmark our models on manually annotated Product Descriptions(PD) test set along with public Indic WAT21 benchmark(Nakazawa et al., 2021) in table 6. We consistently show better BLEU scores on all test sets than Public translation API(Google).

We define the Translation Accuracy i.e., the rate at which the translation is acceptable with only minor errors(percent excluding bad cases), is very high across all languages. This allows us to de-

ploy the Translation Systems in large-scale, highly precise settings. Table 7 shows the exact figures for manual evaluation for English to Hindi catalog translations. Our models show remarkably low bad translation cases and very high, (> 50%) gold standard translations. The huge domain gap between e-commerce and general domains leads to poor evaluation results for Google as it produces consistent non-colloquial words and is not adapted to the domain.

The training pipeline has consistently shown better translation models throughout the model iterations paired with Active Learning, adding more monolingual data and filtered high-quality synthetic parallel translations. As evident from the plot 4, the addition of more synthetic data in pre-training, the addition of forward translation for pre-training as well as domain adaptation, model re-training from scratch with higher quality corpus and better pipeline sub-modules, and active learning steps show very significant improvements at each stage. Starting from 32.3 BLEU score, we have reached 41.32 BLEU score, which is a massive improvement just using a few active learning steps and synthetic corpus updates.

4.1 Deployment and Business Impact

Currently the Translation models for all the languages are deployed in batch-prediction mode on CPU inference system. While translating the catalog data or updating the translation models, we trigger the deployment pipeline and update the offline batch-predictions in the database.

The primary metric used to determine the impact of this deployment is conversion and cost savings. We have seen +11 bps improvement in conversion and significant cost savings through 100% automated translations via our system across various

languages.

5 Conclusion

In this work we have shown that synthetic parallel corpus generation and data filtering is a viable option to train large-scale translation models in low-resource settings. Also we show that Active Learning can consistently improve the model. We build very robust, large-scale models which work very precisely on our In-domain data and also outperform Google on public general domain benchmarks consistently. We also show how building colloquial models are important for ease of understanding, and we also show that our overall approach and best practices extend well to multiple Indian languages.

Limitations

The proposed training pipeline heavily relies on synthetic translations. In some cases (for example, Assamese has only <1M monolingual text), there is not enough data, and the initial model itself can not be appropriately trained, which makes the entire pipeline ineffective. Data efficiency is a considerable challenge in low-resource settings.

The pipeline uses several Language Model based sub-modules for data-cleaning, translation quality estimation, etc., which also impact the pipeline capability, and it might get cumbersome to manage and update many modules.

References

- Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. [Towards a Cleaner Document-Oriented Multilingual Crawled Corpus](#). *arXiv e-prints*, page arXiv:2201.06642.
- Loïc Barrault, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, Shervin Malmasi, Christof Monz, Mathias Müller, Santanu Pal, Matt Post, and Marcos Zampieri. 2019. [Findings of the 2019 conference on machine translation \(WMT19\)](#). In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 1–61, Florence, Italy. 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](#).
- Kamal Gupta, Dhanvanth Boppana, Rejwanul Haque, Asif Ekbal, and Pushpak Bhattacharyya. 2021. [Investigating active learning in interactive neural machine translation](#). In *Proceedings of Machine Translation Summit XVIII: Research Track*, pages 10–22, Virtual. Association for Machine Translation in the Americas.
- Aizhan Imankulova, Takayuki Sato, and Mamoru Komachi. 2019. [Filtered pseudo-parallel corpus improves low-resource neural machine translation](#). *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19:1–16.
- Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc’Aurelio Ranzato. 2017. [Unsupervised machine translation using monolingual corpora only](#).
- Toshiaki Nakazawa, Hideki Nakayama, Isao Goto, Hideya Mino, Chenchen Ding, Raj Dabre, Anoop Kunchukuttan, Shohei Higashiyama, Hiroshi Manabe, Win Pa Pa, Shantipriya Parida, Ondřej Bojar, Chenhui Chu, Akiko Eriguchi, Kaori Abe, Yusuke Oda, Katsuhito Sudoh, Sadao Kurohashi, and Pushpak Bhattacharyya, editors. 2021. [Proceedings of the 8th Workshop on Asian Translation \(WAT2021\)](#). Association for Computational Linguistics, Online.
- Álvaro Peris and Francisco Casacuberta. 2018. [Active learning for interactive neural machine translation of data streams](#). In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 151–160, Brussels, Belgium. Association for Computational Linguistics.
- Jerin Philip, Shashank Siripragada, Vinay P. Namboodiri, and C. V. Jawahar. 2021. [Revisiting low resource status of indian languages in machine translation](#). In *8th ACM IKDD CODS and 26th COMAD*. ACM.
- Martin Popel, Marketa Tomkova, Jakub Tomek, Łukasz Kaiser, Jakob Uszkoreit, Ondřej Bojar, and Zdeněk Žabokrtský. 2020. [Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals](#). *Nature Communications*, 11(4381):1–15.
- Gowtham Ramesh, Sumanth Doddapaneni, Aravindh Bheemaraj, Mayank Jobanputra, Raghavan AK, Ajitesh Sharma, Sujit Sahoo, Harshita Diddee, Mahalakshmi J, Divyanshu Kakwani, Navneet Kumar, Aswin Pradeep, Srihari Nagaraj, Kumar Deepak, Vivek Raghavan, Anoop Kunchukuttan, Pratyush Kumar, and Mitesh Shantadevi Khapra. 2021. [Samanantar: The largest publicly available parallel corpora collection for 11 indic languages](#).
- Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. [Leveraging pre-trained checkpoints for sequence generation tasks](#). *Transactions of the Association for Computational Linguistics*, 8:264–280.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. [Improving neural machine translation models with monolingual data](#).

- Chau Tran, Yuqing Tang, Xian Li, and Jiatao Gu. 2020. [Cross-lingual retrieval for iterative self-supervised training](#).
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).
- Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2020. [CCNet: Extracting high quality monolingual datasets from web crawl data](#). In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.
- Boliang Zhang, Ajay Nagesh, and Kevin Knight. 2020. [Parallel corpus filtering via pre-trained language models](#).

Topic Modeling by Clustering Language Model Embeddings: Human Validation on an Industry Dataset

Anton Eklund
Umeå University
Adlede AB
Umeå, Sweden

anton.eklund@cs.umu.se

Mona Forsman
Adlede AB
Umeå, Sweden

mona.forsman@adlede.com

Abstract

Topic models are powerful tools to get an overview of large collections of text data, a situation that is prevalent in industry applications. A rising trend within topic modeling is to directly cluster dimension-reduced embeddings created with pretrained language models. It is difficult to evaluate these models because there is no ground truth and automatic measurements may not mimic human judgment. To address this problem, we created a tool called STELLAR for interactive topic browsing which we used for human evaluation of topics created from a real-world dataset used in industry. Embeddings created with BERT were used together with UMAP and HDBSCAN to model the topics. The human evaluation found that our topic model creates coherent topics. The following discussion revolves around the requirements of industry and what research is needed for production-ready systems.

1 Introduction

Contextual advertising is a rising solution for ad placement on the Internet, which avoids the need for user data and cookies. However, to find good contexts for a placement, the content of a page needs to be known and classified as a useful advertising context. News media are dependent on advertising for funding their work and is therefore an important market for contextual advertising. The news is constantly changing, which makes it difficult to create classifiers that can catch and categorize new articles. A possible way to solve this is to use unsupervised topic models, which are powerful tools to structure large collections of text data.

Traditional approaches to do topic modeling are stochastic, the most well-known one being Latent Dirichlet Allocation (LDA) (Blei et al., 2003). The problem with stochastic approaches is that they are slow and getting increasingly more difficult to integrate with modern language models (Zhao et al., 2021; Vayansky and Kumar, 2020). To tackle this,

Neural Topic Models (NTMs), which leverage the power of neural networks to create topic models, are becoming increasingly popular. The techniques of our particular interest are Neural Topic Modeling by Clustering Embeddings (NTM-CE). We define NTM-CE as models that use a distance-based clustering algorithm on the document embeddings created with a Pretrained Language Model (PLM). Performing topic modeling by directly clustering embeddings has been shown to perform comparably to LDA (Sia et al., 2020), or better (Meng et al., 2022), and is claimed to create more coherent topics than other types of NTMs (Zhang et al., 2022). These results make the models attractive for use within industry as analytical tools, and hopefully as part of an automatic classification pipeline. Here, we evaluate a modified BERTopic (Grootendorst, 2022) on an industry dataset consisting of unstructured news articles from a brief period of time.

The evaluation of topic models is not trivial since the lack of annotated datasets makes methods like the F1 score not applicable. It also is counterintuitive to our purpose of having flexible topic models if they are evaluated through static dataset categories. Instead, the field has gravitated to automatic measurements which do not require a ground truth like the Normalized Pointwise Mutual Information (NPMI) (Bouma, 2009). NPMI is a popular way to evaluate topic models, as it is claimed to emulate human judgment of topic coherence (Lau et al., 2014). However, an alarming study by Hoyle et al. (2021) argues that automatic evaluation with NPMI cannot emulate human judgment, a fact which topic modeling papers usually rely on to make their claims. From an industry perspective, it is important to be able to trust and validate topic models before they can be used in a production system. Additionally, to the best of our knowledge, there are no studies that use human evaluation for NTM-CE. Therefore, this paper presents a human expert evaluation using our novel

tool Systematic Topic Evaluation Leveraging Lists of ARticles (STELLAR), which is described further in Section 4. The human expert evaluation is described in Section 5.

A problem that remains for topic models, including NTM-CE, is the interpretability of the resulting topics. This is usually addressed by selecting a set of keywords deemed to be the most descriptive of the topic. The words closest to the centroid of a cluster can be used as descriptors as seen in Bianchi et al. (2021). Another solution by Grootendorst (2022) is to use a class-based term weighting to extract keywords. The question remains if, and to what extent, human evaluators would find the keywords descriptive enough for the overall topic. Hence, we add an assessment of the topic description using a simple four-point scale.

In this paper, we demonstrate NTM-CE in the industry setting of contextual advertisement and do a human expert evaluation of the topic model using our new STELLAR tool. The NTM-CE is an implementation of BERTopic, described in Section 3, that has been applied to a news data set, described in Section 2. The STELLAR tool for topic evaluation is described in Section 4, with further explanation of the human expert evaluation in Section 5. The results of the human evaluation are presented in Section 6, with further discussions of the process, the results, and future improvements in Section 7.

2 Data

The dataset used for this study is a unique collection of publicly available English online news articles. The collection consists of 10000 articles from 58 publishers collected between 2022-05-29 and 2022-06-22. The lengths of the articles range from 501 to 99000 characters with a mean length of 3052. 9753 articles are shorter than 10000 characters. Except for removing articles shorter than 500 characters, no other filtering of the articles was applied. This makes the dataset have the same characteristics as the news from the sampled weeks, with topics such as the *Queen’s jubilee*, *Cancelled flights*, and *Formula 1 racing* taking up a disproportionately large part of the content. These are examples of large but brief news topics that will be irrelevant in a few weeks, illustrating the dynamic nature of the news cycle and why static classifiers are of limited use.

3 Topic Modeling Pipeline

Our NTM-CE approach adopts the pipeline of BERTopic (Grootendorst, 2022) and CETopic (Zhang et al., 2022) by using the sequence presented in Figure 1. The class-based TF-IDF of Grootendorst (2022) is used to create keywords for the topics. The components are described in more detail in the following section.

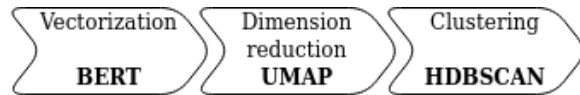


Figure 1: The topic modeling pipeline starts with vectorization using BERT, followed by dimension reduction using UMAP, and ends with clustering using HDBSCAN.

Vectorization was performed with Transformer-based (Vaswani et al., 2017) model BERT (Devlin et al., 2019). It was chosen as it has been widely used in neural topic models and shown to perform well (Grootendorst, 2022; Zhang et al., 2022; Bianchi et al., 2021). Those models used SentenceBERT from Reimers and Gurevych (2019). However, to have the results less tied to a specific BERT model, the model in this project used the HuggingFace base model¹ (768D, 12A, 12L) which was fine-tuned with Masked Language Modeling for the task. As the final document embedding we used the averaged token embeddings of *hidden_state 1*.²

Dimension reduction of high dimensional vectors is used to, among other things, reveal patterns in the data and reduce vector space noise. Techniques for dimension reduction come in two main categories: dimensionality reduction based on matrix factorization and based on neighbor graphs. In this study, we used the neighbor graph method UMAP (McInnes et al., 2018) because it was reported to be both faster and have better clustering quality than the popular t-SNE (Maaten and Hinton, 2008) in the original article.

Clustering has a plethora of techniques but we settled for HDBSCAN (Campello et al., 2013;

¹https://huggingface.co/docs/transformers/model_doc/bert

²Using the unconventional *hidden_state 1* as the embeddings was due to a bug in the code which was found after the human evaluation was completed. However, the vector space is similar to the embeddings from the more conventional *last_hidden_state*. Therefore, for showcasing STELLAR, and exploring NTM-CE, we deemed using the embeddings from *hidden_state 1* sufficient as the topic model still follows our definition of NTM-CE.

McInnes and Healy, 2017) because of its successful use in Grootendorst (2022) and its ability to dynamically choose the number of topics and their size. We used soft clustering for HDBSCAN, meaning that all points in the vector space get assigned to a cluster, which in turn means that no points were considered outliers.

4 The STELLAR Topic Browser

Systematic Topic Evaluation Leveraging Lists of ARTicles (STELLAR) is a tool developed to simplify the in-depth exploration of a topic model into what constitutes the topics rather than just considering the top keywords. The user wants to: 1) get a visual overview of what topics exist and how they are related to each other, 2) be able to quickly identify articles that do not fit into the topic, and 3) go beyond keywords to validate a topic. To solve 1), there is a list of topics with their description alongside a 3D vector space visualization. For 2) and 3), the proposed solution is to allow the user to read the title and keywords of the article and, if needed, to read the text body. The tool needs to be dynamic and interactive, as the user needs the flexibility to study topics freely and investigate different aspects without recreating the topic model and plots.

STELLAR, shown in Figure 2, was created as an application that can run directly in an Internet browser. Its purpose was to aid the user to perform activities 1–3 as described above. It was implemented using the Python Flask³ library. The core functionalities are a topic list, an article list from the chosen topic, a box for the article text body, and a 3D visualization made in Plotly⁴. For each topic, the articles can be marked as not belonging to the topic and thus assist with the evaluation of the topics. The 3D visualization shows the individual articles as points in the vector space reduced to three dimensions, which are color-coded to the cluster to which they belong. Hovering over the articles shows the title and the keywords of the article which helps the user to get a better understanding of the cluster and search in different sections of a cluster. The repository⁵ for STELLAR was released.

³<https://flask.palletsprojects.com/en/2.1.x/>

⁴<https://plotly.com/python/>

⁵<https://github.com/antoneklund/STELLAR>

5 Human Expert Evaluation

The first human evaluation of the topic model and STELLAR was made by three experts (including the first author) in the field of news space analysis. From now we call them *evaluators*. An evaluator is distinguished from an annotator in this work, by having the more complex task of contextualizing a set of articles, finding patterns, and drawing the line for what is considered a topic by excluding articles. In contrast, an annotator selects topics from a list of choices and makes decisions for individual articles. We deem the evaluation task too complex to easily be crowd-sourced. We acknowledge that three evaluators may be too few to make strong claims about NTM-CE in general. However, for our purpose of demonstrating one NTM-CE model on an industry dataset, as well as collecting suggested improvements for STELLAR, we deemed the small expert group adequate.

We make a distinction between the terms *cluster*, *topic*, and *focus topic*. A *cluster* is a set of points that are grouped by the clustering algorithm representing a group of article embeddings. A *topic* is a cluster of article embeddings combined with the descriptive topic keywords. This is the output of the topic model. A *focus topic* is the topic of a cluster that the evaluators decide that most of the article supports.

Each evaluator received an individual dataset with 20 randomly sampled articles per topic. Five of the articles per topic overlapped between the evaluators to calculate inter-rater agreement. The task given was to systematically analyze each topic by reading the article titles and keywords, and reading the article body if needed. Then, record their evaluation by 1) deciding on a focus topic with the help of suggestions⁶, 2) record the id of articles that did not belong to the focus topic, and 3) give a score between 1 and 4 on how well they thought the keywords given by the topic model reflected the focus topic. The scores correspond to: 1=very bad, 2=bad, 3=good, and 4=very good. We chose a four-point scale to force the evaluators to decide if the keywords are good or bad. The instructions given to the evaluators are specified in Appendix A.

Inter-rater agreement AG was used to assess the

⁶The list of suggested topics was compiled by the first author which will introduce biases. However, evaluators were encouraged to record their custom focus topic if none of the items on the list was satisfactory.

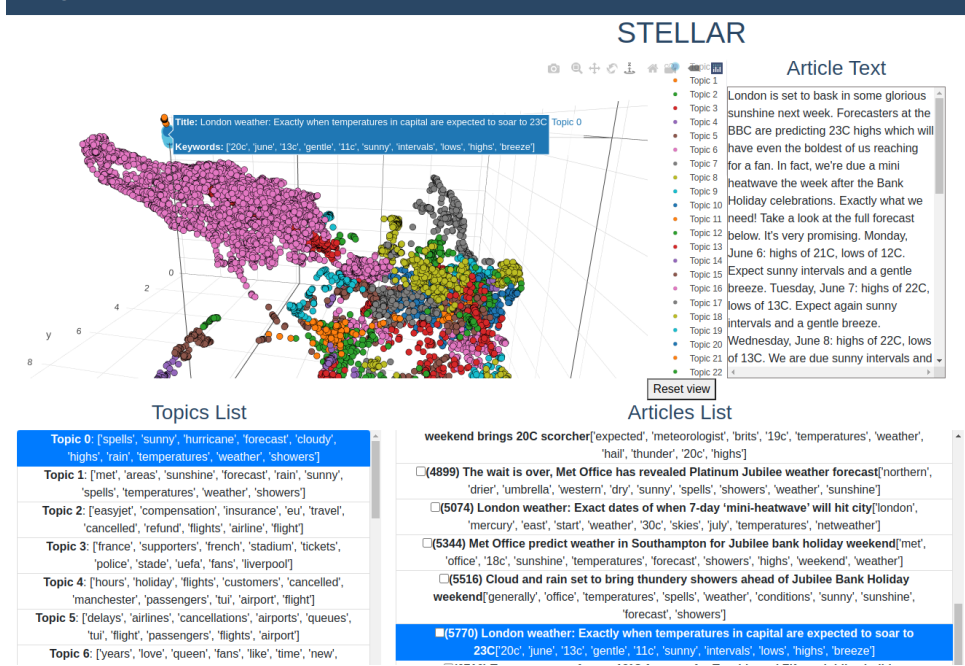


Figure 2: The user interface of STELLAR. The four core components are the 3D visualization, a list of topics with keywords, a list of articles from chosen topics with keywords, and the article text of a chosen topic.

reliability of the evaluators. It was calculated as:

$$AG = \frac{n_{agree}}{N} \quad (1)$$

where n_{agree} is the number of the agreeing decisions, and N is the number of possible decisions. Two different types of decisions are aggregated. The first type of decision is the focus topic of the clusters, called *Agreement focus topic*. The second type of decision is for each overlapping article. The evaluator decides whether they belong to the focus topic or not, called *Agreement overlapping articles*.

To assess whether the topic model produced coherent topics. Our definition of evaluator-determined coherence score (Coh) is:

$$Coh = 1 - \frac{n_{misplaced}}{n_{articles}} \quad (2)$$

where $n_{articles}$ is the number of articles evaluated in the topic and $n_{misplaced}$ is the number of articles that the evaluators found was misplaced into that topic. We call a topic *coherent* if $Coh \geq 0.8$. This threshold at 80% is where we consider a topic coherent enough for being useful in an industrial application. Further, a topic that has at least one evaluator labeling it *Incoherent* will be considered incoherent, regardless of the opinions of other evaluators.

We are aware that the judgment of how coherent a topic is will depend on the individual experiences

Nr topics found	63
Nr coherent topics	52
Average <i>Coh</i> for coherent topics	96%
Articles in coherent topics	57%
Agreement focus topic (including incoherent topics)	87%
Agreement focus topic for coherent topics	98%
Agreement overlapping articles	95%
Keywords describing topic	2.8

Table 1: Statistics of the topic modeling and the human evaluation.

and interests of every evaluator. However, the purpose of the evaluation is not to find a ground truth of what is the focus topic, but rather to determine if the articles presented by the model form a coherent topic. If the evaluators draw the line on what constitutes a topic differently, we see that as a limitation of the model, and the reduction in coherence score is justified.

6 Results

The topic modeling pipeline resulted in 63 clusters with varying sizes as seen in Figure 3. The largest one contains 3347 articles, and the smallest ones are around 20. In total, 2367 of the 10000 articles were manually analyzed. The evaluators agreed on 95% of their decision on overlapping

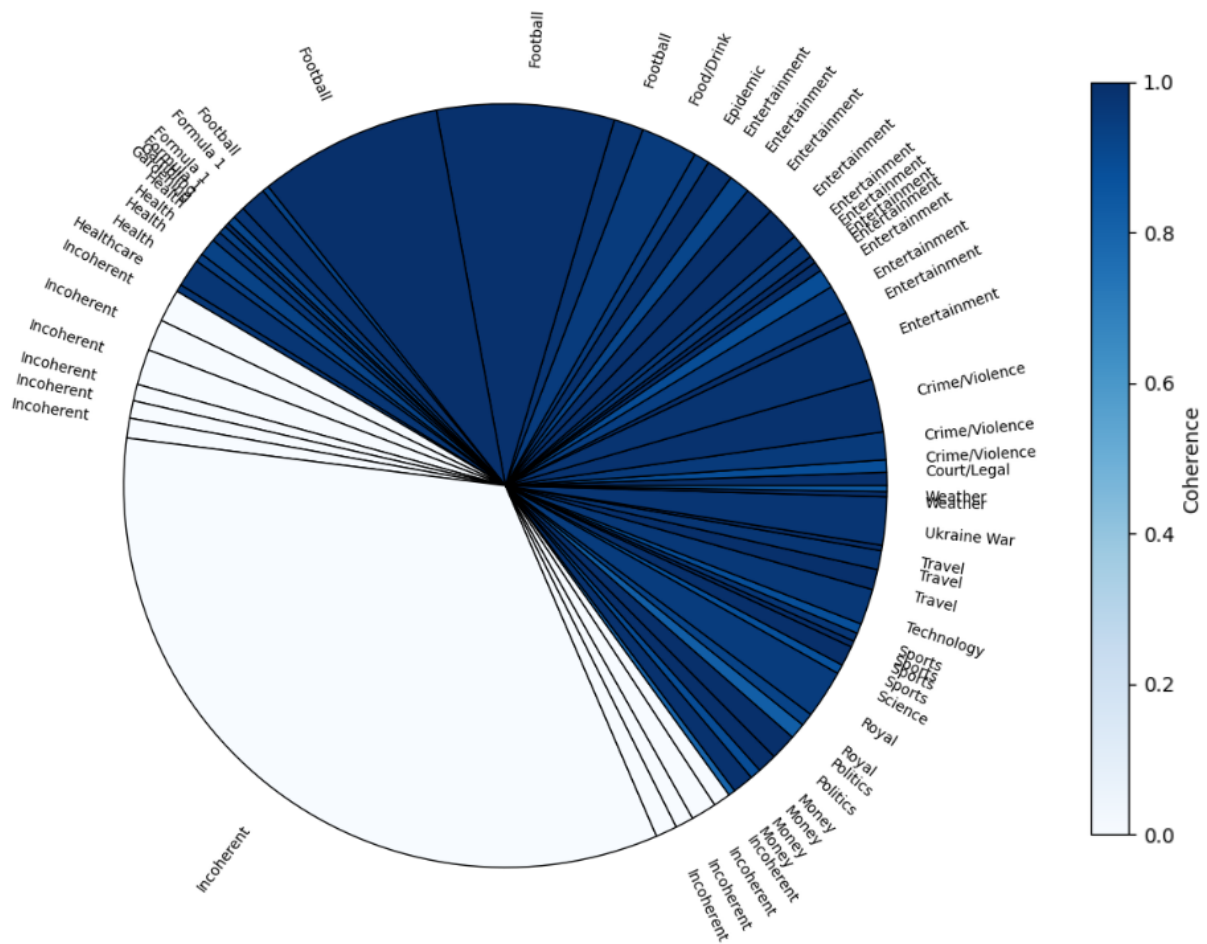


Figure 3: A pie chart illustrating the sizes and coherence of all topics. Light blue means that the topic is incoherent. Different shades of blue indicate how strong the coherence is based on the human evaluation. In the north-west part of the graph, the mixed wedges are Formula 1, Gambling, and Gardening.

articles. The focus topics identified by the evaluators were [*Court/Legal*, *Crime/Violence*, *Entertainment*, *Epidemic*, *Food/Drink*, *Football*, *Formula 1*, *Gambling*, *Gardening*, *Health*, *Incoherent*, *Money*, *Politics*, *Royal*, *Science*, *Sports*, *Technology*, *Travel*, *Ukraine War*, *Weather*], a total of 20 focus topics. A focus topic can be assigned to multiple clusters. Broader focus topics such as *Entertainment* contain articles about gossip, celebrities, movies, and TV-series. The *Sports* topic is all sports excluding *Football* and *Formula 1*. It is made up of tennis, combat sports, golf, cricket, and rugby among the larger ones.

The data from the human evaluation in Table 1 shows that 52 out of 63 topics were over 80% coherent. Topics that were determined coherent had an average coherence score of 96%. Those topics contain 5653 of the articles, which is 57% of the dataset. A little less than half of the articles ended up in incoherent topics. The largest *Incoherent* cluster (see Figure 3) consists of shorter articles,

with an average length of 1500 characters. The features of *Incoherent* clusters will be further explored in Section 7.

The evaluators agree on the focus topic for 87% of topics. In topics where coherence is high, the evaluators agree on the focus topic for 98% of the topics, that is, all topics except one. The disagreements between the evaluators usually came from when one evaluator had chosen *Incoherent* while the others had specified a topic.

Another common disagreement, important for understanding the difficulty of the topic modeling problem is shown in Table 2. The focus topic was about *Weather*, but one can find that the topic consisted of two subtopics, which we can call *Forecasts* and *Hurricanes*. One evaluator decided the focus topic to be *Forecasts* and then continued to mark the articles about *Hurricanes* as misplaced. However, the other evaluators considered the focus topic as *Weather* and thus fully coherent. Cases like these, where one evaluator creates a more nar-

row focus topic than the others, make up for much of the reduction in the total coherence score.

7 Discussion

The topic model found 63 topics in which 20 focus topics were identified. We interpret it as a good partitioning of the corpus, except for the fact that the largest topic was labeled as *Incoherent*. The optimal number of topics found may vary greatly between corpora and the aim should not be to find one cluster per focus topic from the focus topic list, e.g. finding 20 clusters for this dataset. However, for an analytical application in the industry, it would be advantageous to have a way to collect clusters with a similar focus topic into a larger collection. Whether that should be done with the vector space distance or with other methods remains to be studied.

The human evaluation determined 52 out of 63 topics to be coherent, with an average coherency of 96%. However, only 57% of the dataset ended up in coherent topics. One reason for this is the strict requirement that all evaluators should agree on the focus topic for a topic to be considered coherent. However, the foremost reason is that the largest topic, which contained 3347 articles, was labeled *Incoherent*.

A deeper inspection of the clusters of incoherent topics gave interesting insights. The largest cluster contained short articles with half the average length as the rest of the dataset. We assume they have been padded before the BERT vectorization and are clustered on the padding artifacts. An informal test to re-cluster this particular cluster was done to see if dividing it into smaller partitions would create coherent clusters. However, these new clusters did not create coherent topics either. Since we found large coherent clusters such as *Football*, as well as small *Incoherent* clusters, we believe that the clusters should not necessarily be as small nor as balanced as possible, balancing being something [Meng et al. \(2022\)](#) emphasized. Rather, we think that the dynamic properties of HDBSCAN could suffice if guided by inputs from suitable vector space statistics, and applied to a well-formed vector space.

Further analysis of the *Incoherent* topics revealed patterns among the articles but not enough to make them coherent. Some of the topics contain articles on multiple focus topics. An example is a topic with the combination *Real estate, Home*

styling, and *Tourist attractions* that all describe nice environments but not in one coherent focus topic. One topic has locally anchored articles, but about different focus topics and locations. One of the topics is dominated by first-person stories, however, the focus topics differ, and hence it was incoherent in the evaluation. The same can be said for a topic with very emotional content. These topics deserve a deeper examination and understanding, as they have themes that have a stylistic or emotional character. Studying this remains future work since it was not included in the evaluator instructions.

According to the keyword evaluation, the keywords describing the topics were on average positive (> 2.5), yet not good (2.8). The overall positive assessment was still slightly unexpected as the perception before the study was that the keywords did not describe the focus topics well. One factor affecting the results might be that the evaluators had a better understanding of what the keywords mean after reading the topic articles and therefore thought the keywords described the articles well. A more focused study on keywords for topic descriptions needs to be done to investigate this. Nevertheless, since the description was not close to very good (4), ways forward might be to find better keyword extractors or other methods to describe the topics. A preferred scenario for our industry purposes is a topic model where we trust that all topics have $Coh \geq 80\%$ and that the topic descriptions are clear enough for a human to determine the focus topic without looking deeply into what articles are in the topic. An ideal scenario would be that we can trust a system to automatically decide the focus topic similar to human judgment.

The evaluators agreed on the focus topic for 87% of the topics and also had an agreement of 95% for overlapping articles. The agreement on the focus topic for coherent topics was almost perfect at 98%, which means that it was almost always recognizable. However, as shown in [Table 2](#), there are difficulties even for humans to determine where to limit a focus topic. Then, can we expect topic models to do that for us? We expect topic models to be able to divide the articles into topics with a focus reasonably well. However, for the contextual advertisement vital finesse of correctly finding narrow or trendy focus topics, a human will still be needed. An important addition for managing contextual campaigns would be the possibility to analyze topics over time in the style of [Blei and](#)

Topic 0	[spells, sunny, hurricane, forecast, cloudy, highs, rain, temperatures, weather,...]
id: 777	1st of 2022, Hurricane Agatha heads for Mexico tourist towns [landfall, millimeters, mazunte, mexico, kph, storm, puerto, oaxaca, center, ...]
id: 2510	Hurricane Agatha is first named storm of Atlantic season after hitting Mexico... [maximum, hurricanes, noaa, atlantic, storm, inches, southern, mexico, hurricane, ...]
id: 2197	Met Office gives Scotland weather update for Queen’s Platinum Jubilee weekend [forecast, places, warmer, rain, unsettled, dry, drier, weather, spells, showers]
id: 4899	The wait is over, Met Office has revealed Platinum Jubilee weather forecast [northern, drier, umbrella, western, dry, sunny, spells, showers, weather, sunshine]
id: 5770	London weather: Exactly when temperatures in capital are expected to soar to 23C [20c, june, 13c, gentle, 11c, sunny, intervals, lows, highs, breeze]

Table 2: Example of when the evaluators disagreed on the focus topic. One evaluator decided the focus topic to be *Forecasts* and then continued to mark the articles about *Hurricanes* (on the top) as errors. However, the other evaluators decided that the whole topic was coherent as *Weather*.

Lafferty (2006) or Wang and McCallum (2006). This is something that Grootendorst (2022) has been working on with BERTopic. Another observation was that it would be beneficial if the time-consuming analysis was only required to be done once and then have systems detecting new topics emerging and disappearing.

The tool STELLAR, presented in this paper, was created to allow an evaluator to systematically evaluate a topic model by reading the articles that make up a topic. The main purpose was to be able to apply a credible coherence score on a topic model while using as little evaluator time as possible. We believe that STELLAR aids this purpose reasonably well. Since this is the first evaluation with STELLAR, naturally, there are improvements to be made both to the evaluation process and the tool itself. When doing human evaluation of topic models, usually the concept of an intrusion task is used to identify how coherent a topic is (Chang et al., 2009; Hoyle et al., 2021). This task is not fully transferable to our concept of coherence. However, we believe the incorporation of ideas around the intrusion task would make STELLAR better.

Finally, we believe that using the PLM not only allows for the topic models to stay relevant when new language models are released but also creates a more interpretable vector space for analysis since one can observe what topics are related to each other with visual inspection. This human expert evaluation of NTM-CE concludes that the technique is viable and has many attractive benefits. However, it has some limitations that need to be addressed before being used to its full potential as an automatic classifier.

8 Conclusions

In this study, we applied Neural Topic Modeling by Clustering Embeddings (NTM-CE) made with BERT on an industry dataset of news articles. Our human evaluation of NTM-CE, done with our novel STELLAR tool, agrees with previous studies of the technique: coherent topics can be created by clustering embeddings from a pretrained language model. However, only 57% of the articles ended up in coherent topics. Inspection of incoherent topics revealed them to consist of multiple focus topics, or have some other emotional or stylistic characteristic. Unraveling the workings of incoherent topics to increase the number of articles in coherent topics shows great opportunity for industry application. With the STELLAR tool, we hope to keep improving on NTM-CE as a promising technique for the future.

Ethical Considerations

The dataset was scraped from public news sites of established publishers. No personal blogs were used. The names of the people who are written in the articles are mentioned as public persons. We do not view this as a privacy infringement. The articles are not redistributed.

We identified that our personal biases have an impact on the outcome of the results. Examples are choosing the list of focus topics or determining when to limit a topic. In practice, those choices in turn might have an effect on what type of topic model is deployed in the end. A consideration could be to include a more diverse group of expert evaluators. A model might be too generalizing and fail to identify topics that are associated with marginalized groups or cultures, leading to technology being catered to a homogeneous majority.

Acknowledgements

We thank Adrian Andreasson and Dusan Mitic for being expert evaluators, Frank Drewes and anonymous reviewers for their insightful input, and the rest of the university and Adlede teams. This Ph.D. student is funded by the Swedish Foundation for Strategic Research, project id ID19-0055.

References

- Federico Bianchi, Silvia Terragni, Dirk Hovy, Debora Nozza, and Elisabetta Fersini. 2021. [Cross-lingual contextualized topic models with zero-shot learning](#). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1676–1683, Online. Association for Computational Linguistics.
- David M. Blei and John D. Lafferty. 2006. [Dynamic topic models](#). In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 113–120.
- David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. [Latent dirichlet allocation](#). *Journal of machine Learning research*, 3(Jan):993–1022.
- Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. *Proceedings of the Biennial GSCL Conference*.
- Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems*, 22.
- 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.
- Maarten Grootendorst. 2022. [Bertopic: Neural topic modeling with a class-based tf-idf procedure](#). *arXiv preprint arXiv:2203.05794*.
- Alexander Hoyle, Pranav Goel, Andrew Hian-Cheong, Denis Peskov, Jordan Boyd-Graber, and Philip Resnik. 2021. [Is automated topic model evaluation broken? the incoherence of coherence](#). In *Advances in Neural Information Processing Systems*, volume 34, pages 2018–2033. Curran Associates, Inc.
- Jey Han Lau, David Newman, and Timothy Baldwin. 2014. [Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality](#). In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 530–539, Gothenburg, Sweden. Association for Computational Linguistics.
- Laurens Van Der Maaten and Geoffrey Hinton. 2008. [Visualizing high-dimensional data using t-SNE](#). *Journal of Machine Learning Research*, 9:2579–2605.
- Leland McInnes and John Healy. 2017. [Accelerated hierarchical density based clustering](#). In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE.
- Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. [Umap: Uniform manifold approximation and projection](#). *Journal of Open Source Software*, 3(29):861.
- Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. [Topic discovery via latent space clustering of pretrained language model representations](#). In *Proceedings of the ACM Web Conference 2022, WWW '22*, page 3143–3152, New York, NY, USA. Association for Computing Machinery.
- 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.
- Suzanna Sia, Ayush Dalmia, and Sabrina J. Mielke. 2020. [Tired of topic models? clusters of pretrained word embeddings make for fast and good topics too!](#) In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1728–1736, 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.
- Ike Vayansky and Sathish Kumar. 2020. [A review of topic modeling methods](#). *Information Systems*, 94:101582.
- Xuerui Wang and Andrew McCallum. 2006. [Topics over time: A non-markov continuous-time model of topical trends](#). In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06*, page 424–433, New York, NY, USA. Association for Computing Machinery.

Zihan Zhang, Meng Fang, Ling Chen, and Mohammad Reza Namazi Rad. 2022. [Is neural topic modelling better than clustering? an empirical study on clustering with contextual embeddings for topics](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3886–3893, Seattle, United States. Association for Computational Linguistics.

He Zhao, Dinh Phung, Viet Huynh, Yuan Jin, Lan Du, and Wray Buntine. 2021. [Topic modelling meets deep neural networks: A survey](#). *arXiv preprint arXiv:2103.00498*.

A Human Evaluation Protocol

Instructions for filling in Table

For each Topic in Topics List:

1. Click on the Topic in the Topic list.
2. Notice the keywords describing the Topic.
3. Read the article titles and keywords.
4. Click on the article to read the body if not clear what the topic is about.
5. Choose the main topic from the list. A topic in the list can be chosen multiple times.
 - (a) If you can't find a topic that includes 50% of the articles, then choose "no topic"⁷.
 - (b) If you don't agree with any of the topics in the list, write "custom" and then in the notes write your custom topic. Please make suggestions so that it is not only my biases determining the categories.
6. Write down the article id for articles that do not belong to the topic.
7. Write on a scale (1-4) if you think the keywords are a good representation of the topic. 1=bad, 2=sort of bad, 3=sort of good, 4=good.

While doing the task. Write notes of interesting things that you reflect over. Also, make general notes about what improvements that can be made to the tool.

⁷We have translated 'No topic' to 'Incoherent' when writing the article.

Author Index

- ., Govind, 332
- Abraham, Nabila, 342
- Afzal, Zubair, 323
- Aguirre, Maia, 148
- Anaby Tavor, Ateret, 228
- Ankit, Ankit, 90
- Antognini, Diego, 1
- Antonius, Gilbert, 216
- Apponsah, Kwesi, 450
- Au, Winnie, 450
- Balazs, Jorge, 155, 179
- Banitalebi-Dehkordi, Amin, 29
- Barut, Emre, 427
- Bavadekar, Shailesh, 531
- Belyaev, Vladislav, 295
- Belz, Anya, 121
- Ben-Shaul, Ido, 273
- Bendersky, Michael, 502
- Bertelli, Luca, 502
- Bettencourt-Silva, Joao, 77
- Biś, Daniel, 408
- Boaz, David, 228
- Bonin, Francesca, 77
- Bourgon, Richard, 389
- Bradford, Melanie, 264
- Buckley, Mark, 358
- Cai, Lianshang, 597
- Cao, Qian, 236
- Cecil, Matt, 468
- Chada, Rakesh, 485
- Chen, Bei, 164
- Chen, Cheng, 494
- Chen, Lei, 400
- Chen, Nancy, 247
- Chen, Weizhu, 558
- Chen, Zhiyu, 367
- Cheng, Zhicong, 597
- Chilimbi, Trishul, 9
- Chin, Peter, 264
- Chiu, Justin, 171
- Chou, Houwei, 400
- Church, Chris, 264
- Collins, Marcus, 468
- Comment, Nicholas, 208
- Danchenko, Pavel, 155, 179
- Das, Anasuya, 342
- Dekeyser, Elizabeth, 208
- del Pozo, Arantza, 148
- Deval, Pandya, 450
- Ding, Yifan, 110
- Ding, Zhuoye, 282
- Do, Quynh, 131
- Dolatabadi, Elham, 450
- Dong, Anlei, 558
- dong, junwei, 570
- Duan, Nan, 558
- Eklund, Anton, 645
- El-Kurdi, Yousef, 461
- Emmadi, Madhuri, 295
- Ernst, Patrick, 155, 179
- Estes, Alex, 468
- Fan, Jun, 597
- Fan, Xing, 90, 408, 485
- Fang, Anjie, 367
- Fang, Biyi, 216
- Feng, Hao, 63
- Fetahu, Besnik, 367, 439
- Ford, Elliot, 629
- Forsman, Mona, 645
- Fu, Xue-Yong, 494
- Fuchs, Gilad, 273
- Fusco, Francesco, 1
- Gabrilovich, Evgeniy, 531
- Gandhi, Apurva, 216
- García-Sardiña, Laura, 148
- Gardiner, Shayna, 494
- Garera, Nikesh, 550, 589, 637
- Garrido Ramas, Jose, 164
- Gaspers, Judith, 131
- Gee, Leonidas, 419
- Ghosh, Debanjan, 304
- Gojayev, Turan, 138
- Gong, Yeyun, 558

Goyal, Abhinav, 589
Grant, Christan, 110
Gu, Simiu, 597
Gueudre, Thomas, 138
Guo, Chenlei, 90, 408, 485
Gupta, Jai, 531
Gupta, Nalin, 427
Gupta, Rahul, 381
Gupta, Saurabh, 408, 485
Gupta, Vivek, 216

Hajebi, Kiana, 381
Hamza, Wael, 18
Hao, Jie, 90, 408, 485
Hasan, Rakeb, 358
Haverty, Lisa, 208
HE, Shiqi, 29
HE, XIAOFENG, 570
Hiranandani, Pooja, 494
Hong, Jenna, 216
huang, jun, 570

Ishola, Bukola, 450

J Kurisinkel, Litton, 247
Jackson, Richard, 629
Jiang, Jiarong, 316
Jiang, Ziyang, 90

Kalaiselvi Bhaskar, Karthik Raja, 450
Kamath, Chaitanya, 531
Kandpal, Chandra Shekhar, 198
Kang, Jaewoo, 629
Kannan, Kawshik, 342
Keivanloo, Iman, 9
Khan Khattak, Faiza, 450
Koleva, Aneta, 358
Kumar, Karun, 295
Kumar, Manoj, 164, 381
Kumar, Rajat, 208
Kumar, Vineet, 228
Kunc, Ladislav, 512

Laskar, Md Tahmid Rahman, 494
Lau, Elaine, 450
Le, Dieu-Thu, 164, 264
Le, Hieu, 264
Lee, Sungjin, 90
Lee, Taesung, 541
Lehnen, Patrick, 131
Lei, Zeyang, 522
Li, Alexander Hanbo, 316
Li, Dan, 323

Li, Erin, 450
Li, Haley, 29
Li, Shuanglong, 522
Li, Xian, 110, 477
Li, Yong, 570
Liang, Yan, 110
Liao, Lingrui, 502
Lin, Jimmy, 295, 389
Lin, Peng, 282
Liu, Xiao, 558
Liu, Xinyue, 427
Liu, Yang, 485
Liu, Ye, 254
Liu, Yinghui, 570
Liu, Ziyang, 63
Long, Bo, 282
Loo, Ellen, 389
López-Fernández, Jacobo, 148
Lu, Jingwen, 558
Lu, Weiyi, 9

Ma, Dehong, 597
Ma, Mian, 282
Majumder, Rangan, 558
Malmasi, Shervin, 367, 439
Mandelbrod, Matan, 273
Manotas, Irene, 189
Mao, Yajie, 295
Matin, Mahan, 389
Méndez, Ariane, 148
Metzler, Donald, 531
Miftahutdinov, Zulfat, 606
misra, amita, 99
Moramarco, Francesco, 121
Morishita, Makoto, 616
Munim, Alif, 450
Murakami, Koji, 581
Muravlev, Vladimir, 606
Murray, Craig, 295

Nagata, Masaaki, 616
Nakayama, Yuki, 581
Nallapati, Ramesh, 316
Narain, Jaswinder, 450
Natarajan, Pradeep, 381, 485
Nayak, Ravindra, 550
Ng, Patrick, 316
Nguyen, Tra My, 216
Nigam, Priyanka, 9
Niu, Jingcheng, 450
Niu, Zheng-Yu, 522
Nosakhare, Ehi, 216

Obadinma, Stephen, 450
Odry, Benjamin, 342
Oikawa, Yuto, 581
Oz, Gokmen, 138

Pandey, Akshat, 295
Pandey, Gaurav, 228
Papadopoulos Korfiatis, Alex, 121
Park, Youngja, 541
Patel, Kanishk, 450
Patil, Amey, 637
Pei, Shermin, 208
peng, haoyuan, 254
Peng, Xia, 236
Perera, Mark, 121
Peris, Charith, 138
Poddar, Lahari, 155, 179
Popescu, Octavian, 189
Poroshin, Vladimir, 629
Potdar, Saloni, 512
Purpura, Alberto, 77

Qi, Haode, 512
Qian, Cheng, 512
Qiao, Lingfeng, 254
Qin, Kechen, 427
Quinn, Jerry, 461

Rabinovich, Ella, 228
Raeesy, Zeynab, 427
Rai, Shruti, 208
Rajagopalan, Sunny, 9
Ramakrishna, Anil, 381
Ramamonjison, Rindra, 29
Raman, Karthik, 502
Rawls, Stephen, 427
Reiter, Ehud, 121
Ren, Bo, 254
Ren, Iris, 450
Rengan, Vishnu, 29
Rigutini, Leonardo, 419
Ringsquandl, Martin, 358
Robertson, Sean, 450
Rokhlenko, Oleg, 367, 439, 468
Rottmann, Kay, 164, 264
Rudzicz, Frank, 450

Samanta, Anupam, 502
Sandiri, Spurthi, 427
Savkov, Aleksandar, 121
Schroedl, Stefan, 381
Sehanobish, Arijit, 342
Sengupta, Sudipta, 316

Serrao, Ryan, 216
Serras, Manex, 148
Shaffer, Irene, 216
Shapira, Ofer, 389
Sheinin, Vadim, 189
Shen, Yelong, 558
Shi, Daiting, 597
Shiah, Yu-Jia, 450
Shimizu, Kanna, 208
Shinzato, Keiji, 171
Shneyderman, Anastasia, 606
Sidhorn, Tania, 450
Sil, Avi, 461
Singh, Anupam, 589
Sohoney, Saurabh, 332
Soltan, Saleh, 18, 485
Sorokin, Daniil, 131
Soto, Victor, 18
Srinivasan, Krishna, 502
Srinivasan, Soundararajan, 216
Staar, Peter, 1
Stowe, Kevin, 304
Su, Chengwei, 427
Sun, Mimi, 531
Sun, Xiaodi, 9
Suzuki, Jun, 616
Szarvas, György, 155, 179

Tan, Lizhen, 138
Tang, Raphael, 295
Tanner, Griffin, 450
Tay, Yi, 531
TN, Shashi Bhushan, 494
Torrioni, Paolo, 419
Tran, Ke, 18
Tran, Vinh, 531
Tresp, Volker, 358
Tsatsaronis, George, 323
Tur, Gokhan, 485
Ture, Ferhan, 295
Tutubalina, Elena, 606

Uma Naresh, Niranjana, 90

Vadakkera Suresh, Gautham, 198
Vedula, Nikhita, 468
Veeragouni, Akash, 439
Venkatapathy, Sriram, 381
Vetterli, Thomas, 389
Vetzler, Matan, 228
Vishwanathan, Asha, 198
Vo, Ngoc Phuoc An, 189

Wang, Ang, 570
Wang, Chaokun, 63
Wang, Cheng, 155, 179
Wang, Chengyu, 570
Wang, Gengyu, 512
Wang, Haifeng, 522
Wang, Jianing, 570
Wang, Jun, 316
Wang, Michael, 450
Wang, Qinlong, 236
Wang, Shirley, 450
Wang, Zhiguo, 316
Wanigasekara, Prashan, 427
Waqar, Muhammad, 450
Warrier, Rajeev, 198
Weber, Verena, 264
Wei, Bencheng, 450
Wei, Pan, 138
Wen, Ji-Rong, 558
Weninger, Tim, 110
Wu, Chen, 254
Wu, Fengtao, 208
Wu, Hua, 522
Wu, Lingfei, 63, 282
Wu, Wenquan, 522
Wu, Yi, 597

Xiang, Bing, 316
Xiao, Jiajie, 389
Xiong, Deyi, 236
Xu, Xinchao, 522
Xu, Yi, 9

Yadav, Vikrant, 323
Yang, Fan, 427
Yang, Gefei, 295
Yang, Liqun, 63
Yang, Yi, 522
Yi, Sheng, 216
Yin, Dawei, 597
yin, di, 254
Yoon, Wonjin, 629
Yu, Timothy, 29

Zalmout, Nasser, 110, 477
Zeng, Belinda, 9
Zhang, Bryan, 99
Zhang, Chao, 522
Zhang, Linhao, 597
Zhang, Sean X., 450
Zhang, Taolin, 570
Zhang, Yong, 29

Zhao, Jie, 367
Zhao, Mengxuan, 304
Zhao, Wayne Xin, 558
Zhong, Yizhen, 389
Zhou, Kun, 558
Zhou, Zirui, 29
Zhu, Xiaodan, 400, 450
Ziyadi, Morteza, 381
Zou, Yanyan, 282
Zugarini, Andrea, 419